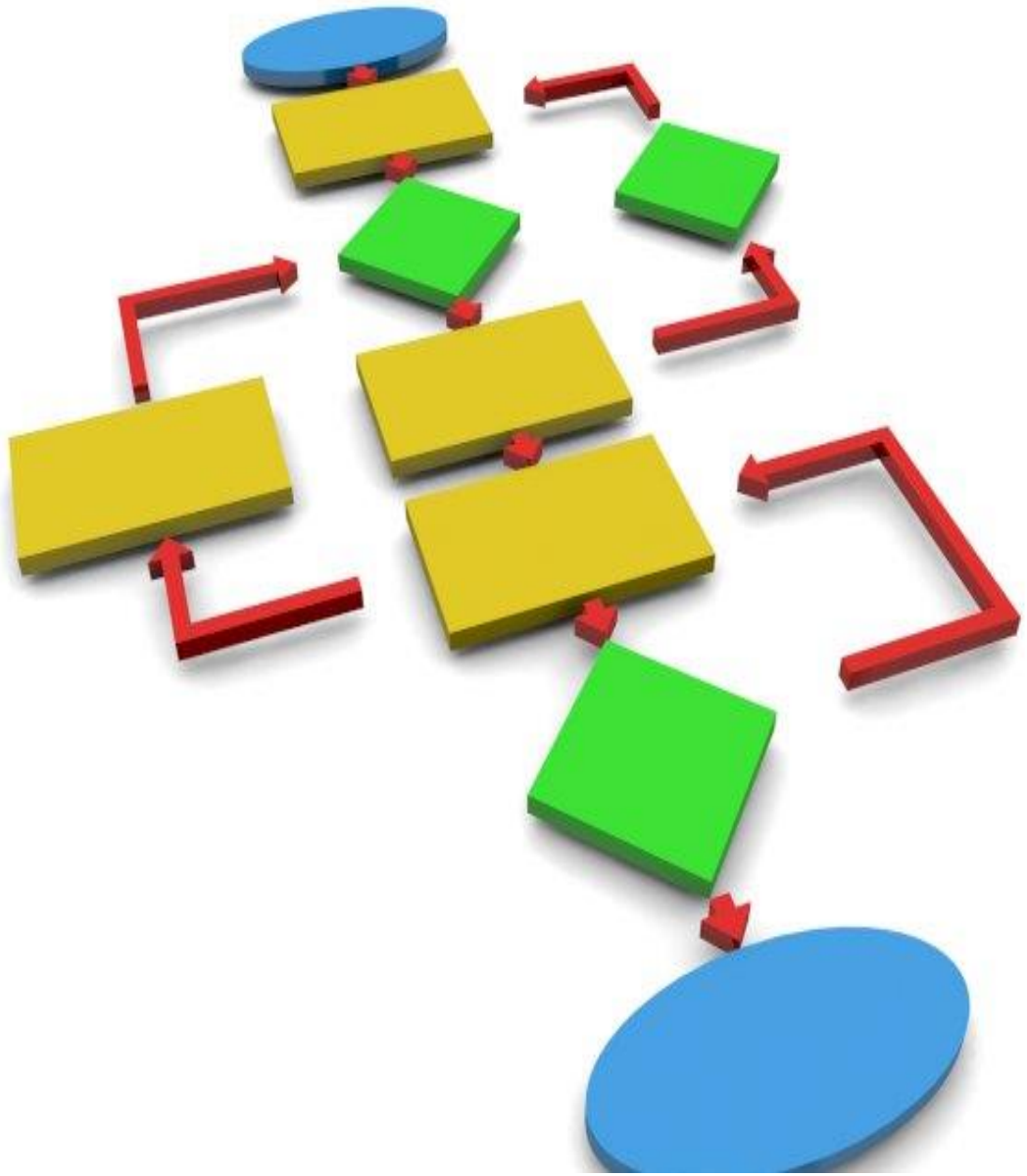
# Project Team - Green

## Phase-2 Training, testing and validation of the final model


**Team Members-**

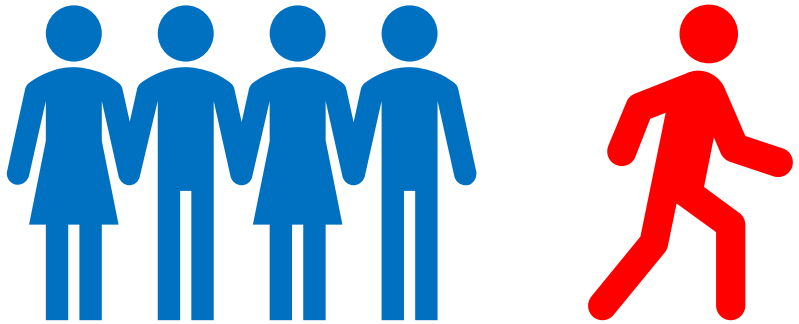
- Yuraja Kadari
- Vicky Jadhav
- Ameya Kalbande
- Yash Jivani



# Agenda

- Goals to achieve
- Model's Selection
- Features Engineering
- Yuraja’s Model
- Vicky’s Model
- Ameya’s Model
- Yash’s Model
- Conclusion

# Goals to Achieve




- The objective of this phase is to select a definitive model capable of accurately classifying records as either Attrited or Existing Customers, while ensuring the required levels of precision and recall.
- This will enable us to make informed business decisions and take appropriate actions to retain valuable customers and reduce attrition rates.
- The chosen model will be the cornerstone of our strategy to optimize customer retention and profitability, and its selection is of paramount importance to the success of our project.

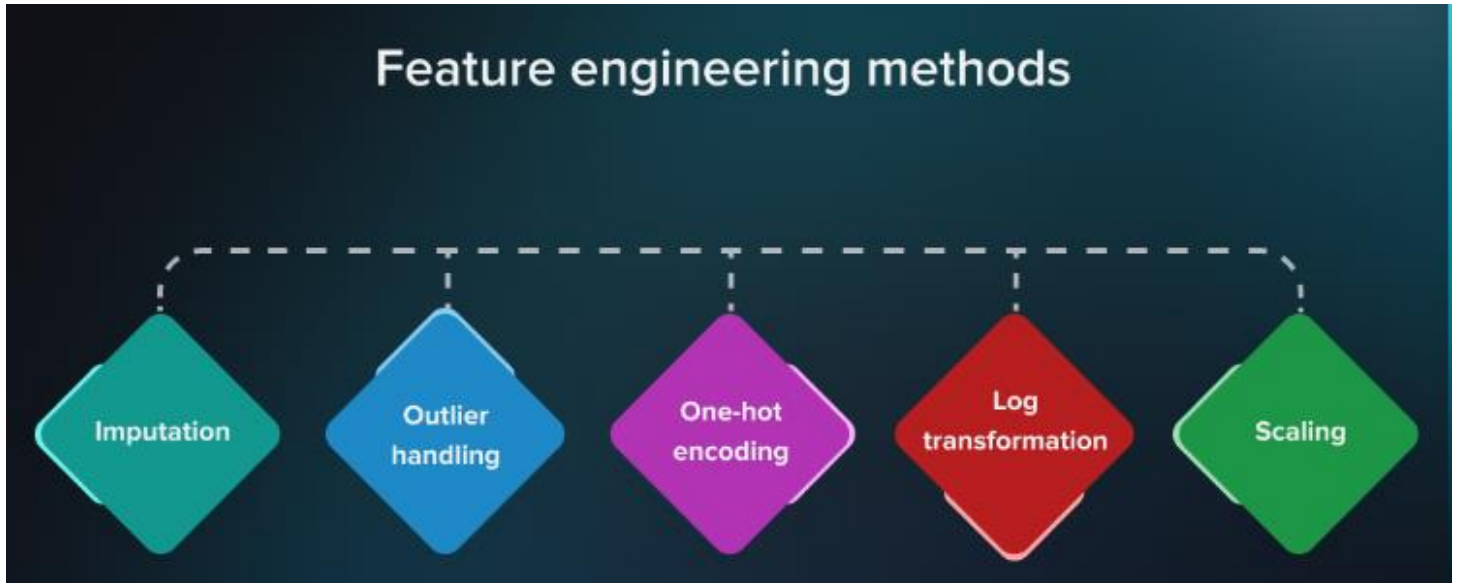
# Model's Selection

- Interpretability: The model should be easily interpretable, meaning that its inner workings and decision-making process can be understood and explained to stakeholders.
- Robustness: The model should be robust to outliers, noise, and missing data. It should also be able to handle changes in the distribution of data over time, which is common in the banking industry.
- Scalability: The model should be scalable, meaning that it can handle larger datasets without sacrificing performance. It's important in the banking industry, where the volume of data is constantly growing.
- Generalization: The model should be able to generalize well to new, unseen data.

Overall, the chosen model should meet multiple criteria, including accuracy, precision, recall, computational complexity, interpretability, robustness, scalability, generalization to ensure that it can make meaningful contributions to reducing customer attrition rates and improving business outcomes in the banking industry.

# Features Engineering

- Scaling and centering numeric predictors
- Removing skewness from numeric variables
- One-hot and dummy variable encoding for categorical variables
- Removing correlated predictors and zero variance variables
- Imputing missing data (more on this later)



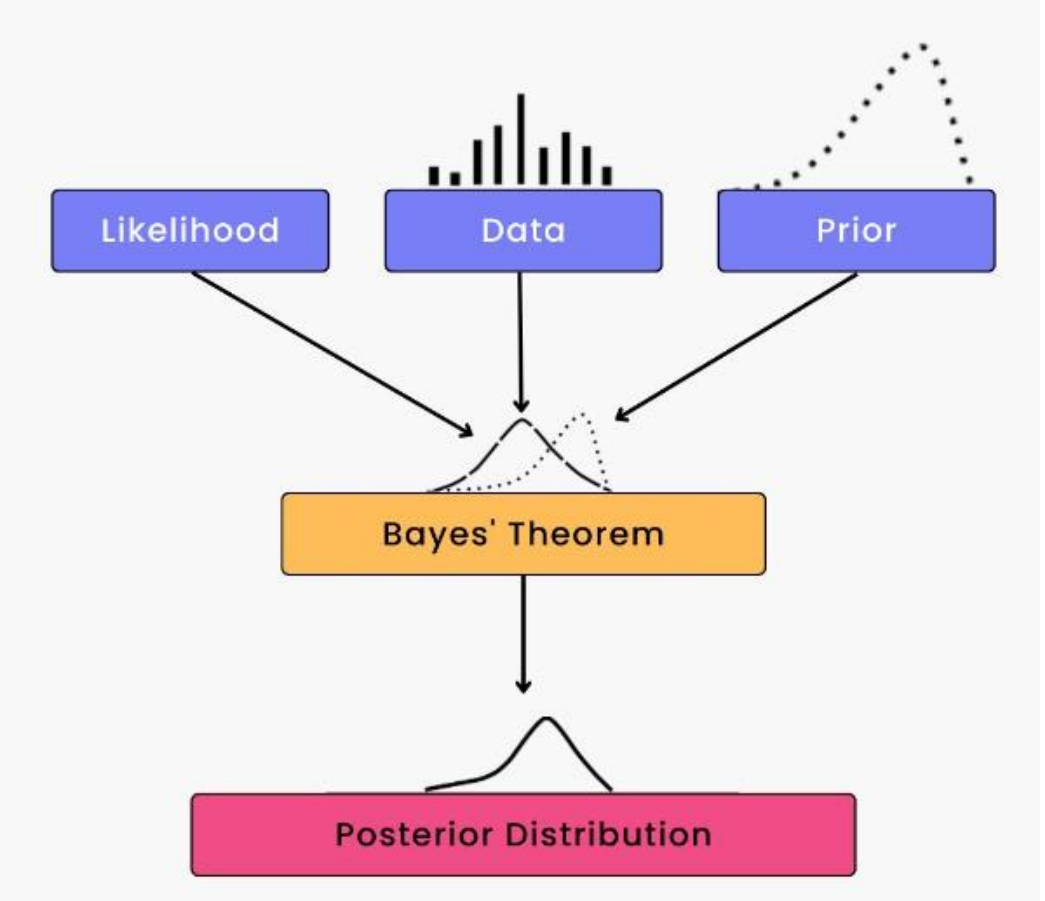
# Naive Bayes Theorem

- Classifiers based on Bayesian methods utilize training data to calculate an observed probability of each class based on feature values.
- When the classifier is used later on unlabeled data, it uses the observed probabilities to predict the most likely class for the new features.
- Basic Concepts (and Notations) of Bayesian Methods:

  • Probability: number of trials in which an event occurred by the total number of trials.

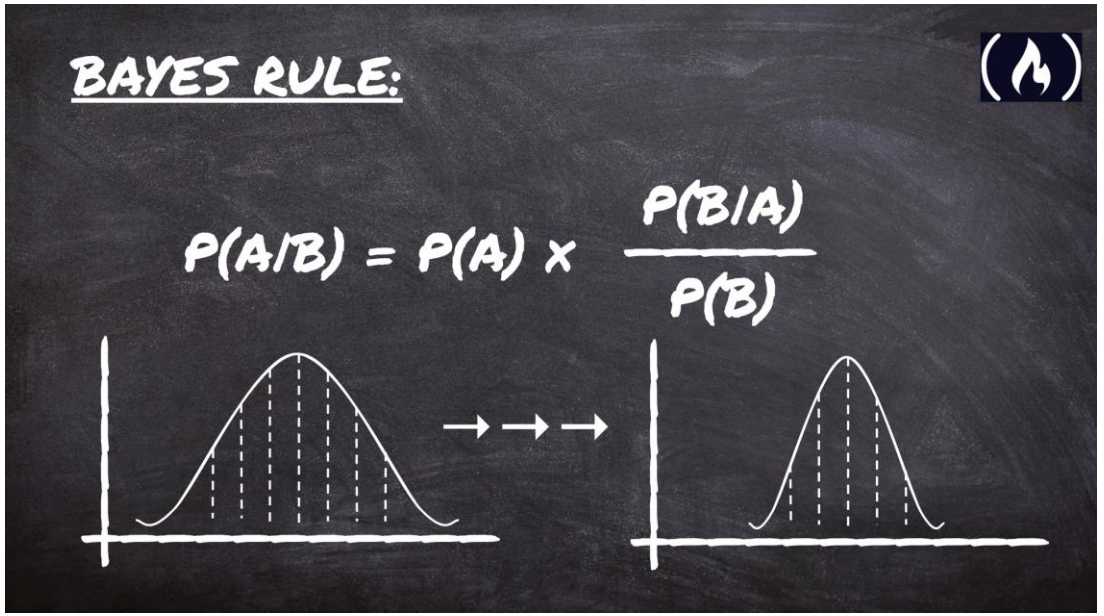
  • Joint probability: the probability of two events happening together given they are independent.

  •Conditional probability with Bayes' theorem: help us calculate the probabilities for dependent events.



# Why Naïve Bayes Theorem ??

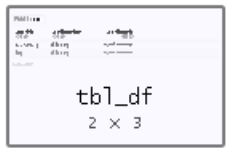
- Naive Bayes is a simple algorithm that is easy to implement and computationally efficient. It is also able to handle large datasets without requiring a lot of computational resources, which is important in the banking industry where the volume of data is often very large.
- Naive Bayes is a probabilistic algorithm that works well with categorical data. In the Bank Churners dataset, there are several categorical features such as Gender, Education Level, and Marital Status that can be used to predict whether a customer is likely to churn or not.
- Naive Bayes is a good choice when the dataset has a relatively small number of features. In the Bank Churners dataset, there are only 21 features, which makes it suitable for Naive Bayes.
- Naive Bayes provides a clear and interpretable output. It calculates the probability of each class given the input data, which can be easily understood and explained to stakeholders in the banking industry.
- Overall, Naive Bayes is a good choice for the Bank Churners dataset because of its simplicity, efficiency, and ability to handle categorical data with a small number of features. It can provide accurate and interpretable results that can be used to reduce customer attrition rates and improve business outcomes in the banking industry.



# NB Model Implementation in R Studio

- Step-1 Data loading
- Step-2 Splitting data into training and testing and stratifying data.
- Step-3 Creating Recipe and applying feature Engineering steps.
- Step-4 Creating working (Preparing and baking Recipe)
- Step-5 Testing the Model

```
nb_model <- naive_Bayes(Laplace = 0.1) %>% set_engine("klaR")

nb_recipe <- recipe(Attrition_Flag ~ ., data = Bankchurners_training) %>%
  step_rm(all_numeric()) %>%
  step_nzv(all_predictors())

nb_workflow <- workflow() %>%
  add_model(nb_model) %>%
  add_recipe(nb_recipe)

nb_fit <- nb_workflow %>% fit(data = Bankchurners_test)
nb_results <- predict(nb_fit, new_data = Bankchurners_test) %>%
              bind_cols(Bankchurners_test %>% dplyr::select(Attrition_Flag))

nb_results$Attrition_Flag <- as.factor(nb_results$Attrition_Flag)

conf_mat(nb_results, truth = Attrition_Flag, estimate = .pred_class)

metrics(nb_results, truth=Attrition_Flag, estimate=.pred_class)
```


A tibble: 2 × 3

| .metric <chr> | .estimator <chr> | .estimate <dbl> |
|---|---|---|
| accuracy | binary | 0.8688889 |
| kap | binary | 0.0000000 |

2 rows

**Accuracy= 86%**

# NB Model Implementation in Jupyter Notebook (Python)

- Step-1 Data loading
- Step-2 Applying Features engineering steps.
- Step-3 Data Splitting (Cross validation)
- Step-4 Applying model
- Step-6 Testing Model

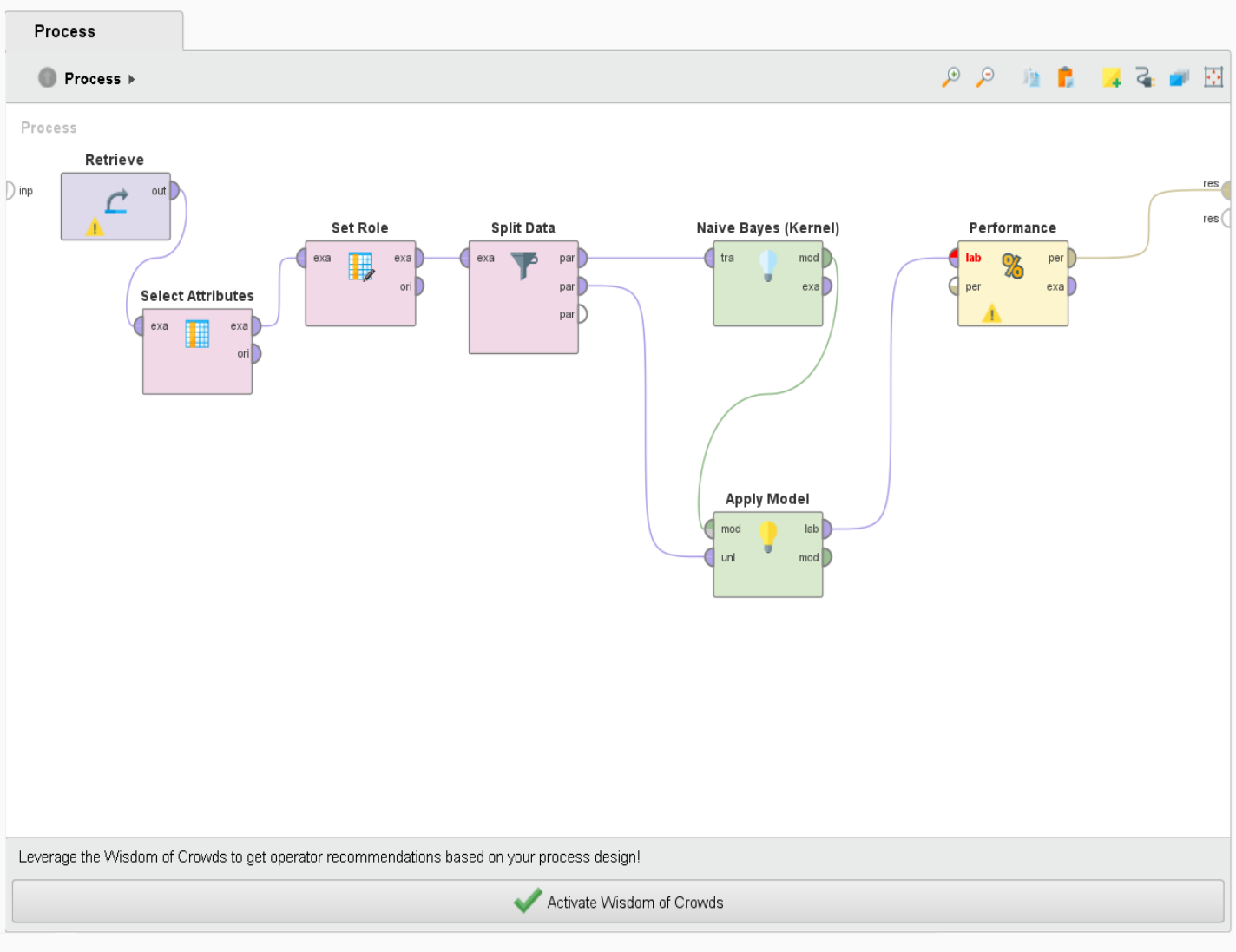
**Accuracy: 92%**
**Precision: 76%**
**Recall: 61%**

In [1]:
```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix, accuracy_score, precision_sc
from sklearn.preprocessing import LabelEncoder

# Load data
bank_churners = pd.read_csv('C:/Users/Lenovo/Downloads/Bank_churners/BankCl
                            usecols=['Gender', 'Dependent_count', 'Months_
                                     'Contacts_Count_12_mon', 'Total_Relat
                                     'Total_Revolving_Bal', 'Total_Trans_C
                                     'Attrition_Flag'])

# Encode Gender column
le = LabelEncoder()
bank_churners['Gender'] = le.fit_transform(bank_churners['Gender'])

# Split data into features and target
X = bank_churners.drop('Attrition_Flag', axis=1)
y = bank_churners['Attrition_Flag']

# Fit Naive Bayes model with cross-validation
nb_model = GaussianNB()
y_pred = cross_val_predict(nb_model, X, y, cv=500)

# Print confusion matrix, accuracy, precision, and recall
confusion_matrix = confusion_matrix(y, y_pred)
accuracy = accuracy_score(y, y_pred)
precision = precision_score(y, y_pred, pos_label='Attrited Customer')
recall = recall_score(y, y_pred, pos_label='Attrited Customer')
print('Confusion Matrix:\n', confusion_matrix)
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
```

```
Confusion Matrix:
 [[ 486  300]
 [ 149 5063]]
Accuracy: 0.9251417139046348
Precision: 0.7653543307086614
Recall: 0.6183206106870229
```

# NB Model Implementation in Rapid Miner

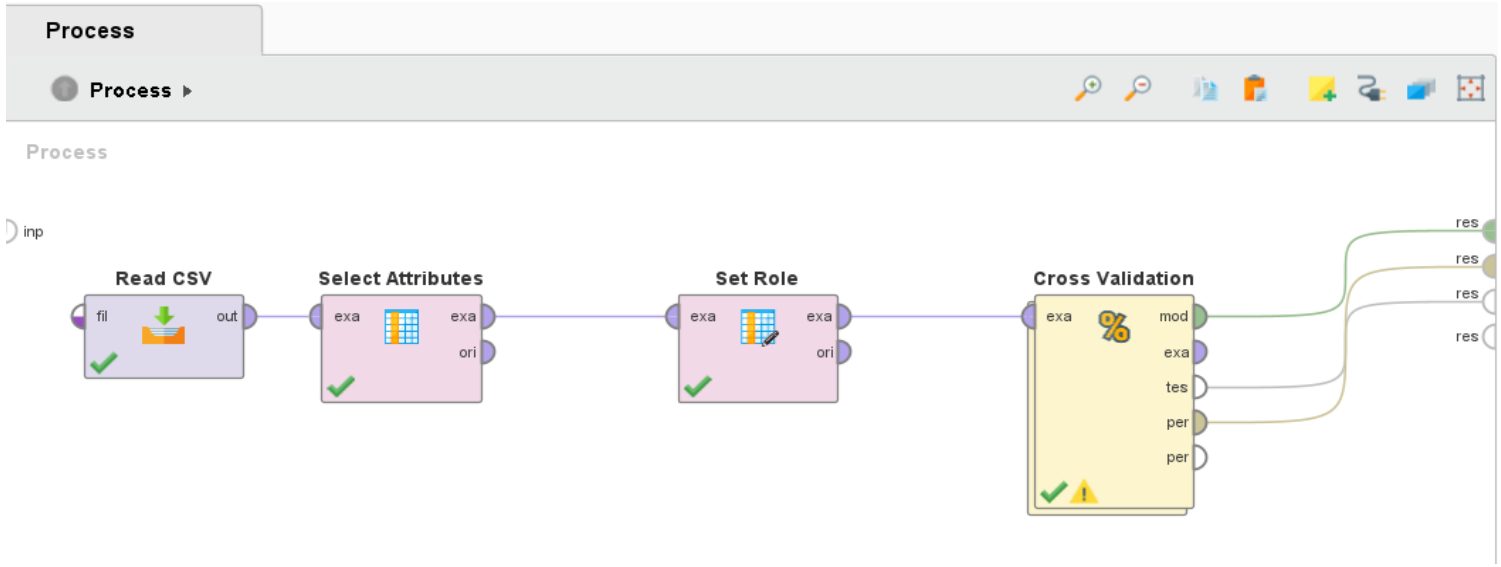
- Step-1 Retrieving Data
- Step-2 Selecting Attributes
- Step-3 Setting roles
- Step-4 Splitting data
- Step-5 Applying Model
- Step -6 Testing model

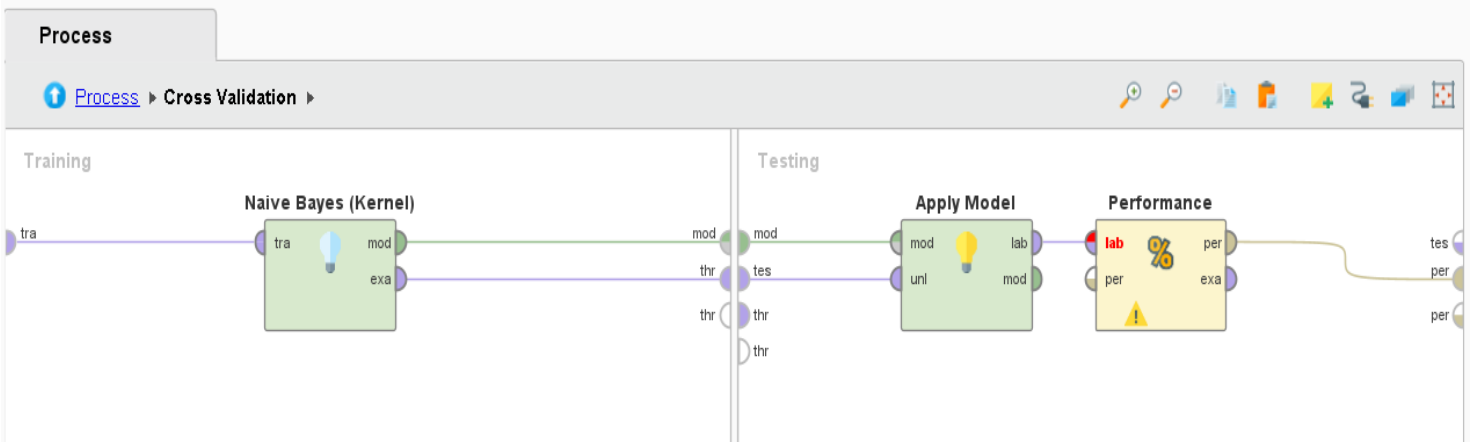



**Accuracy: 90%**
**Precision: 75%**
**Recall: 43%**

# NB Model Implementation in Rapid Miner using Cross Validation

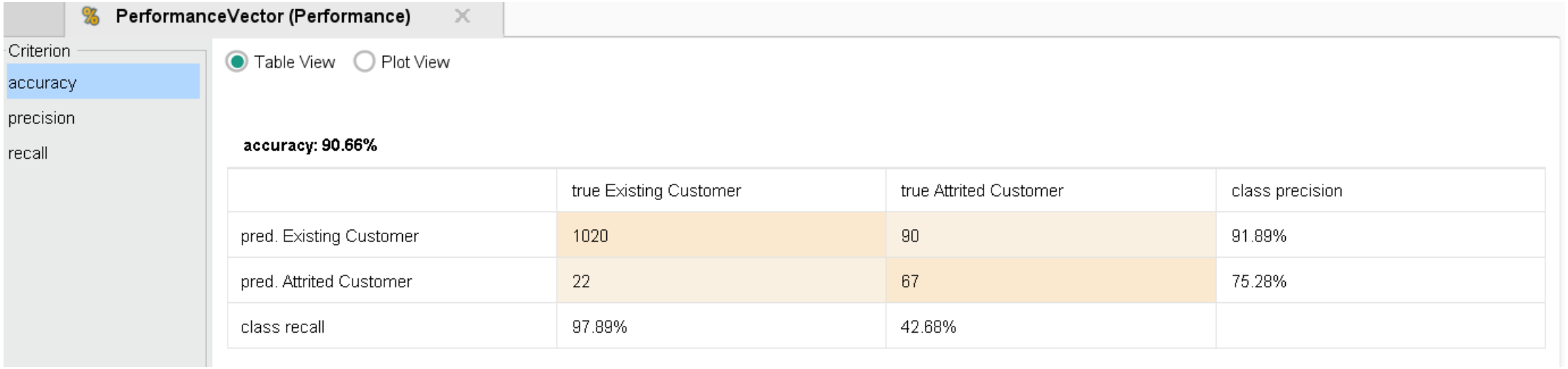
- Step-1 Data Loading
- Step-2 Selecting Attributes
- Step-3 Setting roles
- Step-4 Applying cross validation
- Step-5 Performance



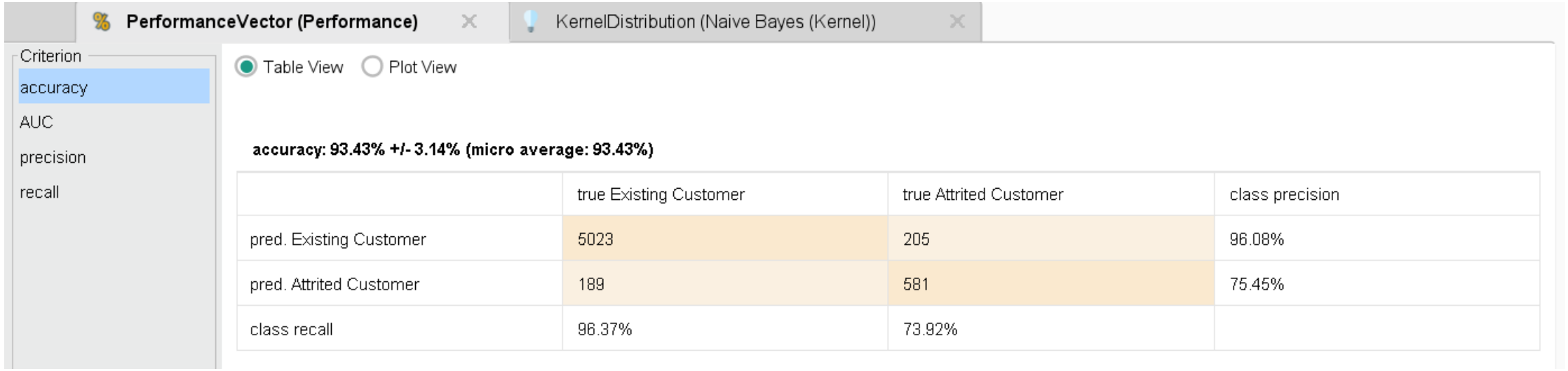

**Accuracy: 93%**
**Precision: 75%**
**Recall: 74%**

- Splitted Data Performance vector




| | true Existing Customer | true Attrited Customer | class precision |
|---|---|---|---|
| pred. Existing Customer | 1020 | 90 | 91.89% |
| pred. Attrited Customer | 22 | 67 | 75.28% |
| class recall | 97.89% | 42.68% | |

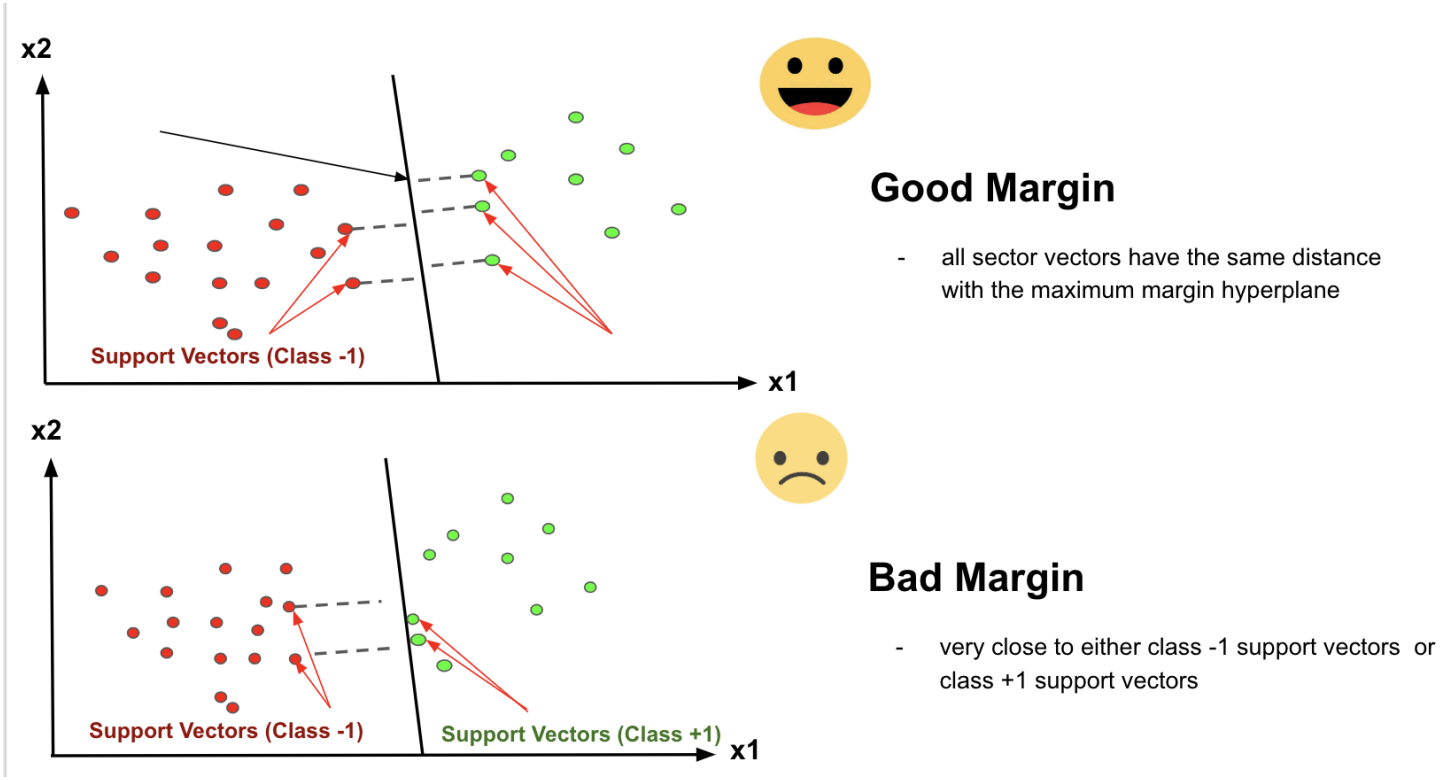
- Cross Validation Performance Vector




| | true Existing Customer | true Attrited Customer | class precision |
|---|---|---|---|
| pred. Existing Customer | 5023 | 205 | 96.08% |
| pred. Attrited Customer | 189 | 581 | 75.45% |
| class recall | 96.37% | 73.92% | |

# What is SVM and Why SVM ??

- SVM stands for Support Vector Machine, which is a type of machine learning algorithm used for classification and regression analysis.
- In classification, SVM finds the best possible boundary (also known as hyperplane) that can separate the different classes in the data. This boundary is chosen such that it maximizes the margin between the two classes, which is the distance between the hyperplane and the closest data points from each class.
- SVM can also handle non-linear boundaries by transforming the data into a higher-dimensional space using a technique called kernel trick.
- SVM can be a good choice for bank churn prediction because it can handle both linear and non-linear boundaries, which can be important for identifying complex patterns in customer data.
- Additionally, SVM is relatively insensitive to the presence of outliers in the data and can handle high-dimensional feature spaces. This can be useful in situations where there are many variables that can affect the likelihood of churn.



# SVM Model Implementation in Jupyter Notebook (Python)

- Step-1 Data Loading
- Step-2 Applying feature engineering steps.
- Step-3 Data Splitting
- Step-4 Performing grid search
- Step-5 fitting model
- Step-6 Testing model

**Accuracy: 93%**
**Precision: 95%**
**Recall: 91%**

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_sc
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# Load data
bank_churners = pd.read_csv('C:/Users/Lenovo/Downloads/Bank_churners/BankCh
                            usecols=['Gender', 'Dependent_count', 'Months_
                                     'Contacts_Count_12_mon', 'Total_Relati
                                     'Total_Revolving_Bal', 'Total_Trans_Ct
                                     'Attrition_Flag'])

# Encode Gender column
le = LabelEncoder()
bank_churners['Gender'] = le.fit_transform(bank_churners['Gender'])

# Split data into features and target
X = bank_churners.drop('Attrition_Flag', axis=1)
y = bank_churners['Attrition_Flag']

# Resample data to balance classes
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size

# Define SVM model
svm_model = SVC()

# Define hyperparameters for grid search
params = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf'], 'gamma': [0.1, 1,

# Perform grid search to find best hyperparameters
grid_search = GridSearchCV(svm_model, params, cv=5)
grid_search.fit(X_train, y_train)
```

```python
# Perform grid search to find best hyperparameters
grid_search = GridSearchCV(svm_model, params, cv=5)
grid_search.fit(X_train, y_train)

# Fit SVM model with best hyperparameters
best_model = grid_search.best_estimator_
best_model.fit(X_train, y_train)

# Make predictions on test set
y_pred = best_model.predict(X_test)

# Print confusion matrix, accuracy, precision, and recall
confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='At
recall = recall_score(y_test, y_pred, pos_label='Attrited
print('Confusion Matrix:\n', confusion_matrix)
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
```

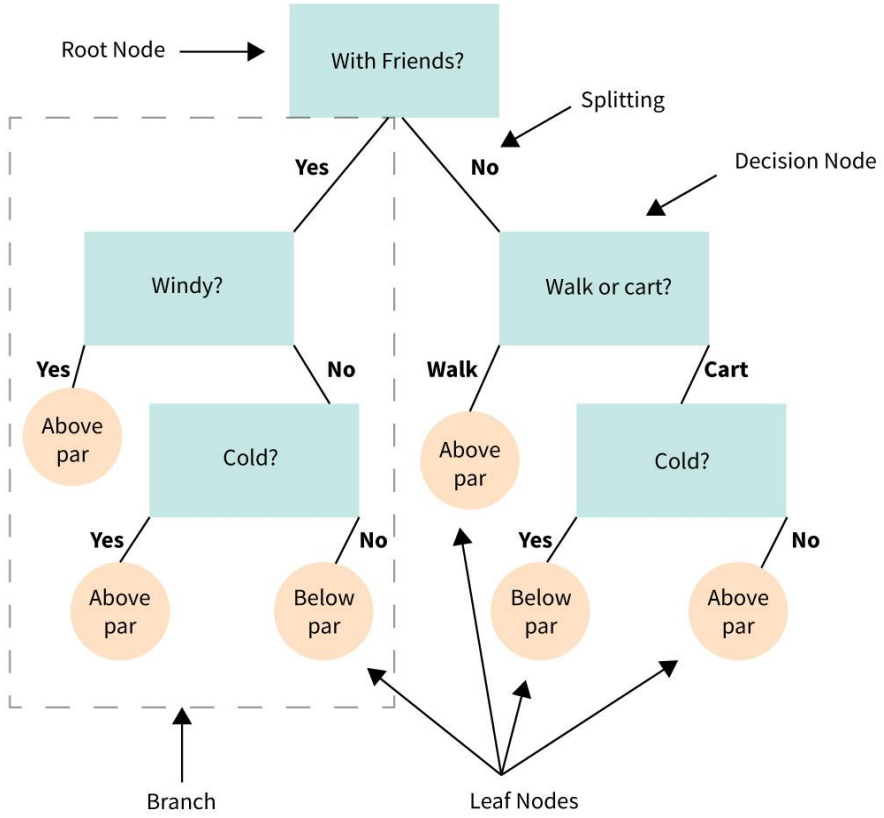
```
Confusion Matrix:
 [[940  98]
 [ 55 992]]
Accuracy: 0.9266187050359712
Precision: 0.9447236180904522
Recall: 0.905587668593449
```

# Summary

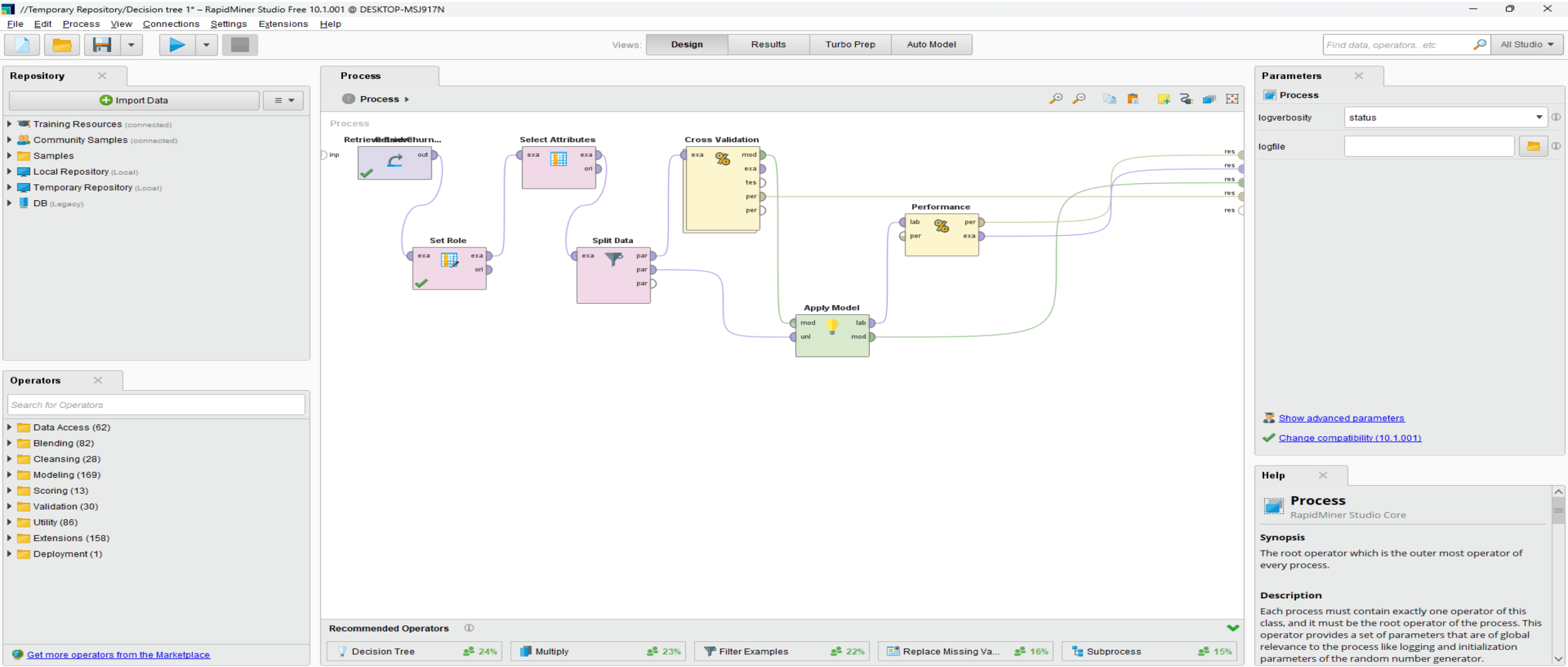
- While selecting the model I first select Naïve bayes but after some time I got stuck because it goes good with more categorical data less numeric data.
- So, I change my model to SVM and got better results shown in above slides.
- My features which I selected in phase 1 shows high co-relation Total trans count, Total Revolving balance.

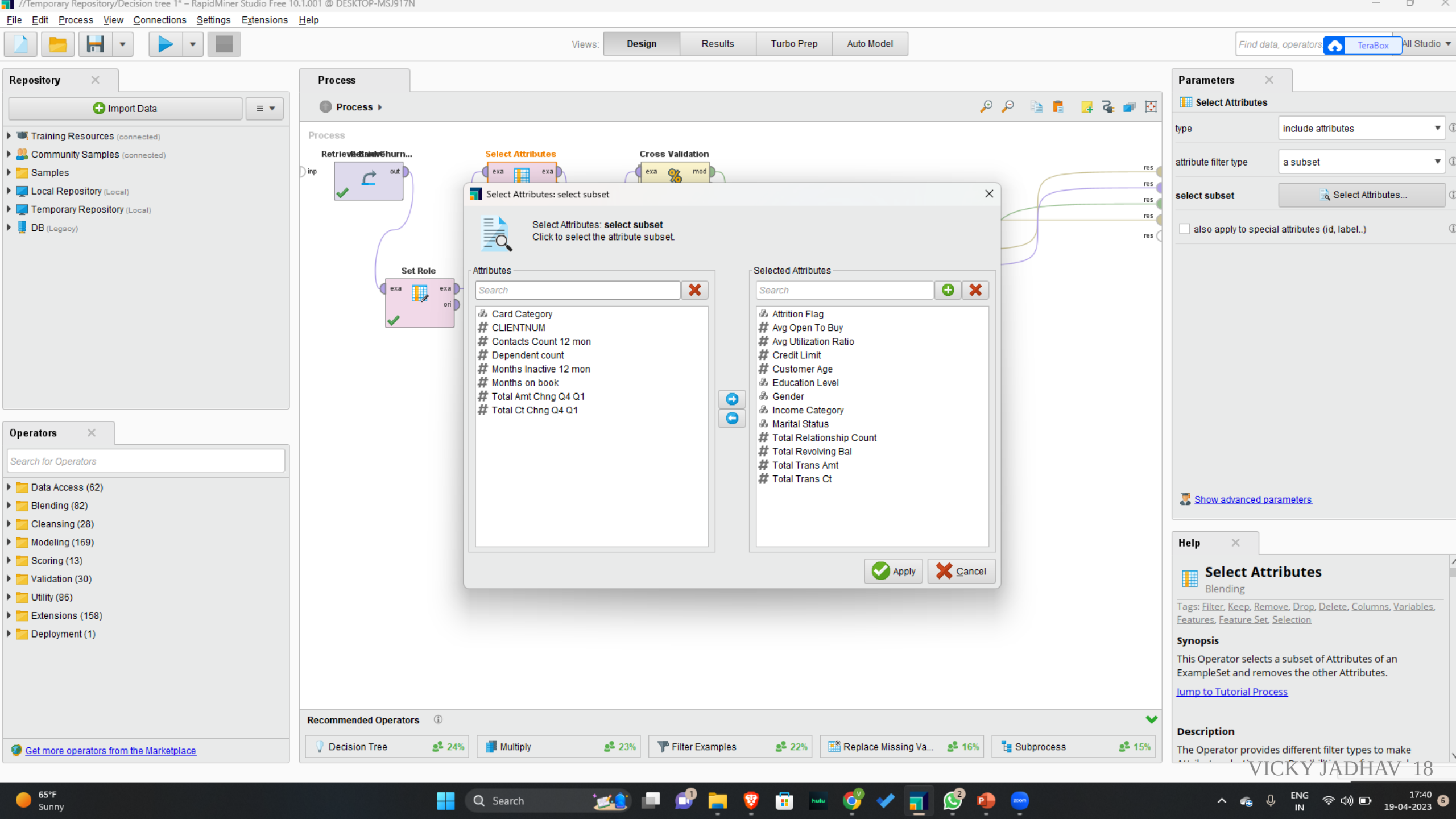
# What is Decision Tree?

- A decision tree is a Supervised Machine Learning Model used to categorize or make predictions based on how previous questions were answered.  A model is a form of supervised learning, meaning that the model is trained and tested on a data set containing the desired categorization.
- The decision tree may not always provide a clear-cut answer or decision. Instead, it may present options so the data scientist can make an informed decision. Decision trees imitate human thinking, so it's generally easy for data scientists to understand and interpret the results.

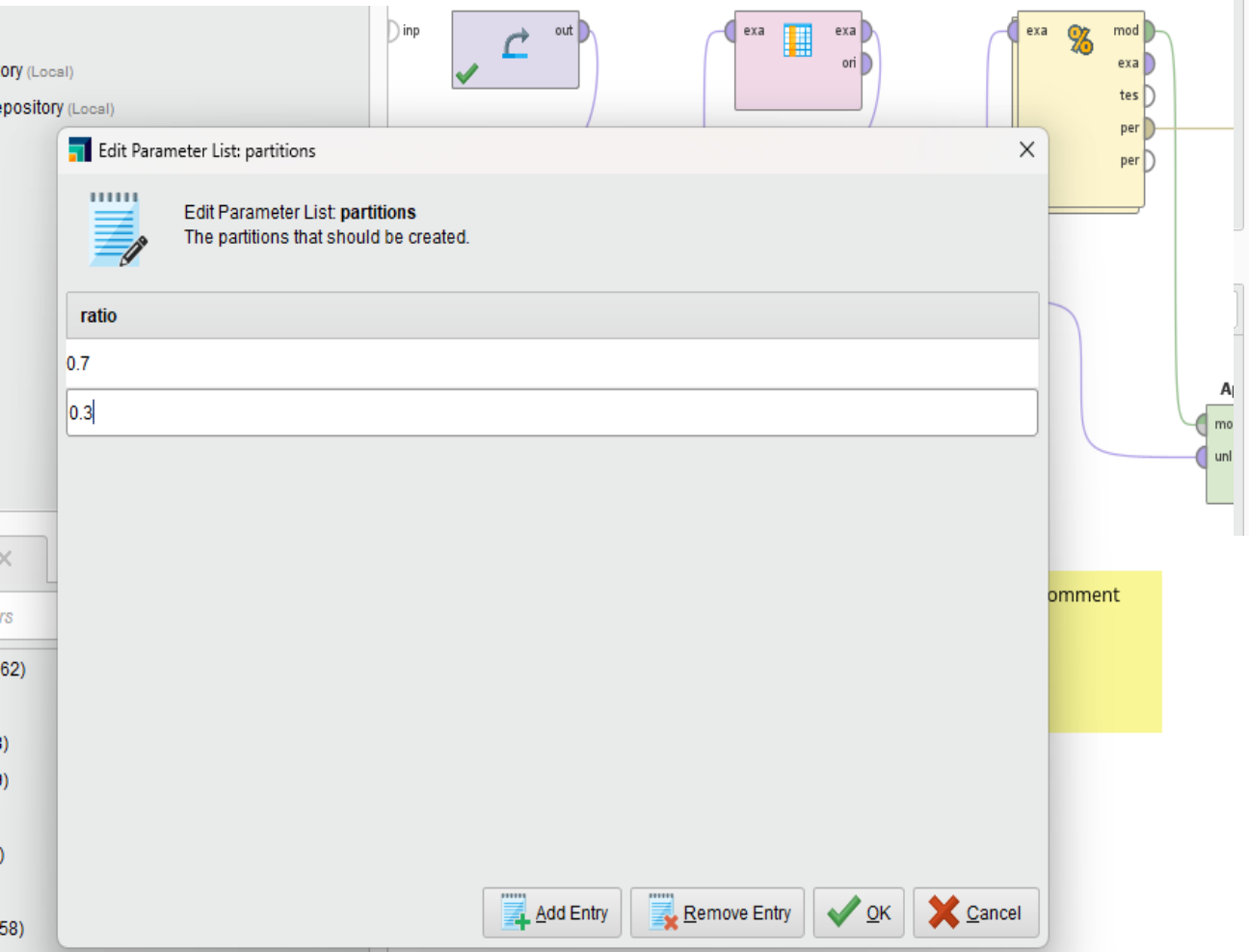


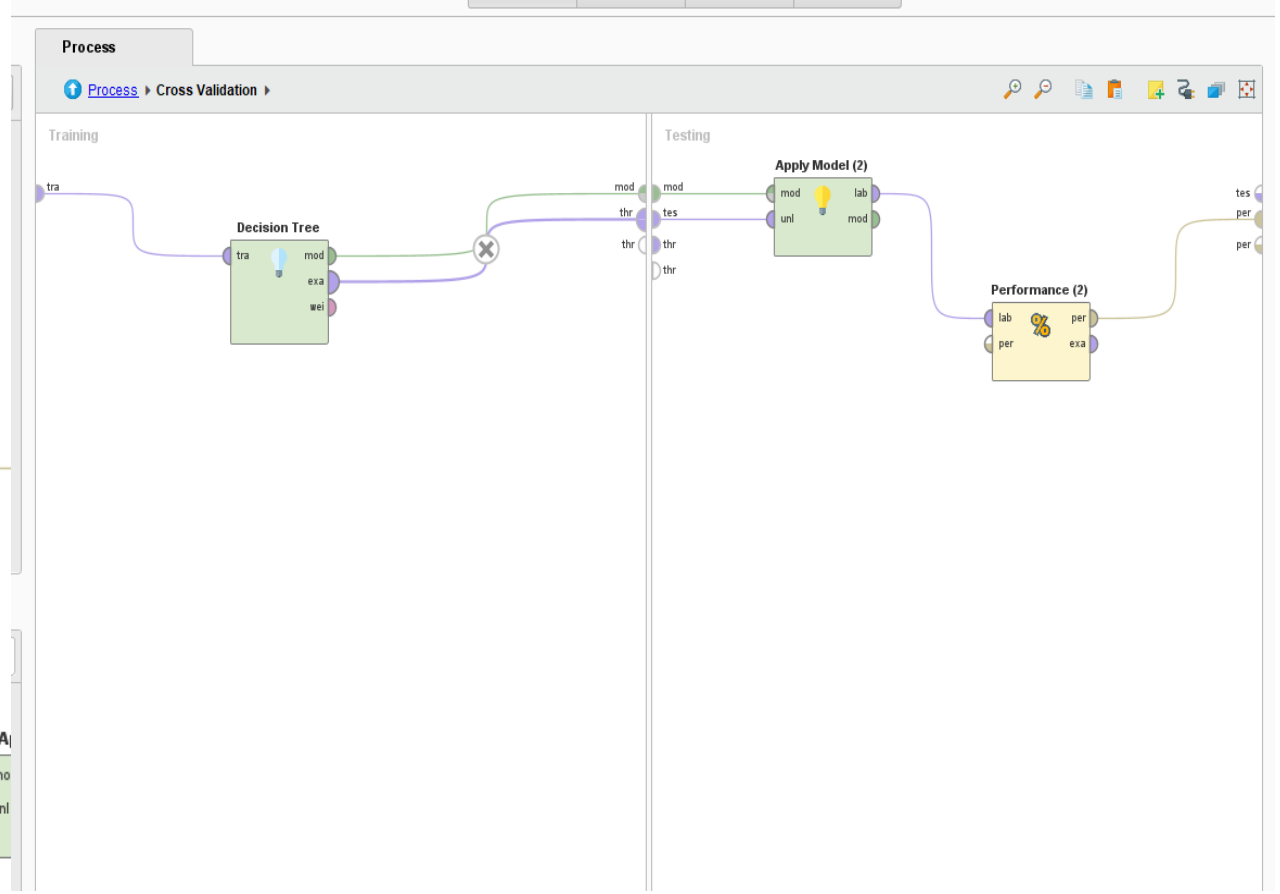
# Model Implementation on RapidMiner




//Temporary Repository/Decision tree 1* – RapidMiner Studio Free 10.1.001 @ DESKTOP-MSJ917N
File Edit Process View Connections Settings Extensions Help
Views: Design Results Turbo Prep Auto Model
Find data, operators TeraBox All Studio
Repository
Import Data
Training Resources (connected)
Community Samples (connected)
Samples
Local Repository (Local)
Temporary Repository (Local)
DB (Legacy)
Operators
Search for Operators
Data Access (62)
Blending (82)
Cleansing (28)
Modeling (169)
Scoring (13)
Validation (30)
Utility (86)
Extensions (158)
Deployment (1)
Get more operators from the Marketplace
Process
Process
Process
Select Attributes
Cross Validation
Set Role
Select Attributes: select subset
Select Attributes: select subset
Click to select the attribute subset.
Attributes
Search
Card Category
CLIENTNUM
Contacts Count 12 mon
Dependent count
Months Inactive 12 mon
Months on book
Total Amt Chng Q4 Q1
Total Ct Chng Q4 Q1
Selected Attributes
Search
Attrition Flag
Avg Open To Buy
Avg Utilization Ratio
Credit Limit
Customer Age
Education Level
Gender
Income Category
Marital Status
Total Relationship Count
Total Revolving Bal
Total Trans Amt
Total Trans Ct
Apply
Cancel
Recommended Operators
Decision Tree 24%
Multiply 23%
Filter Examples 22%
Replace Missing Va... 16%
Subprocess 15%
Parameters
Select Attributes
type include attributes
attribute filter type a subset
select subset Select Attributes...
also apply to special attributes (id, label..)
Show advanced parameters
Help
Select Attributes
Blending
Tags: Filter, Keep, Remove, Drop, Delete, Columns, Variables, Features, Feature Set, Selection
Synopsis
This Operator selects a subset of Attributes of an ExampleSet and removes the other Attributes.
Jump to Tutorial Process
Description
The Operator provides different filter types to make
VICKY JADHAV 18
65°F Sunny
Search
ENG IN
17:40 19-04-2023

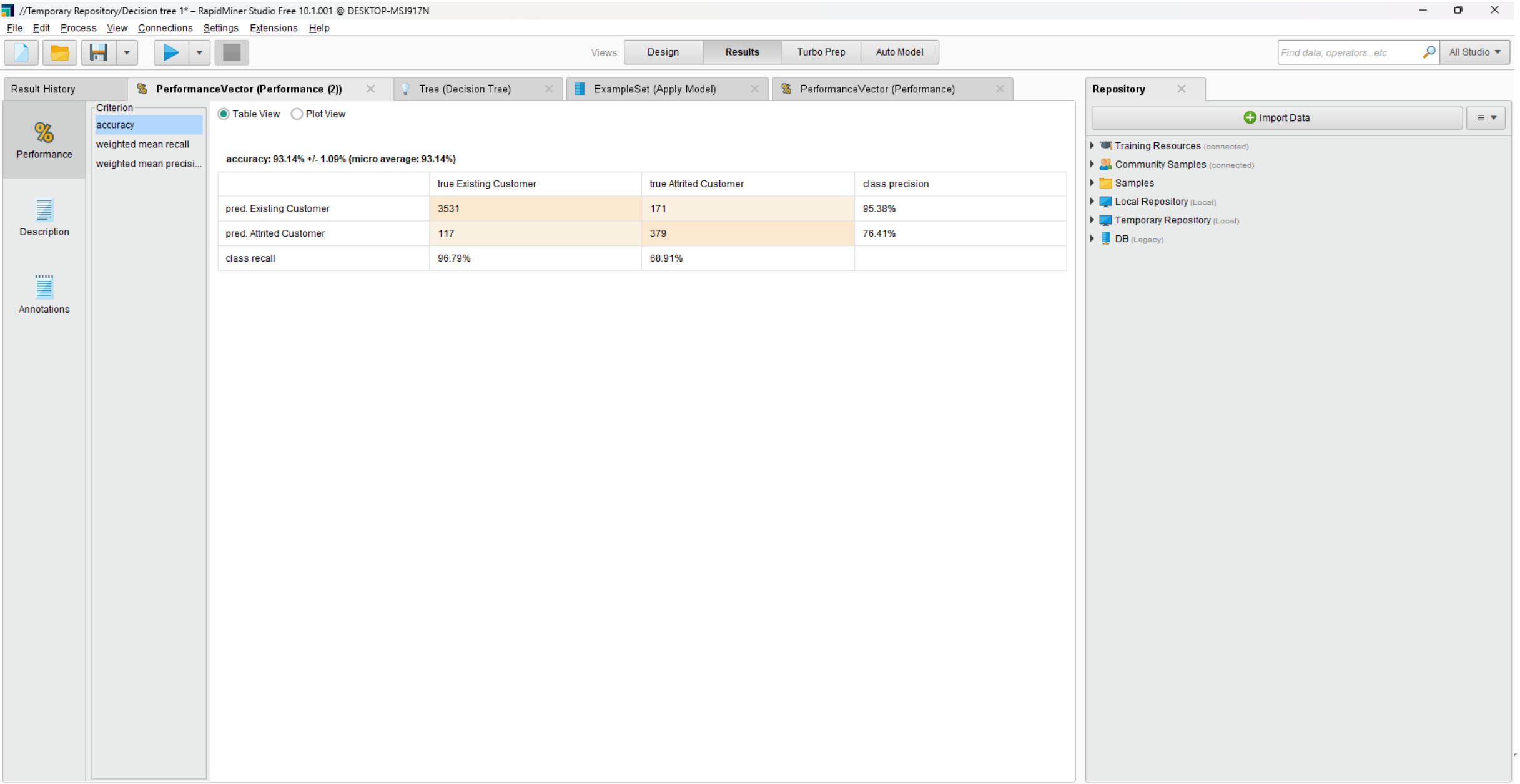
# SPLIT DATA



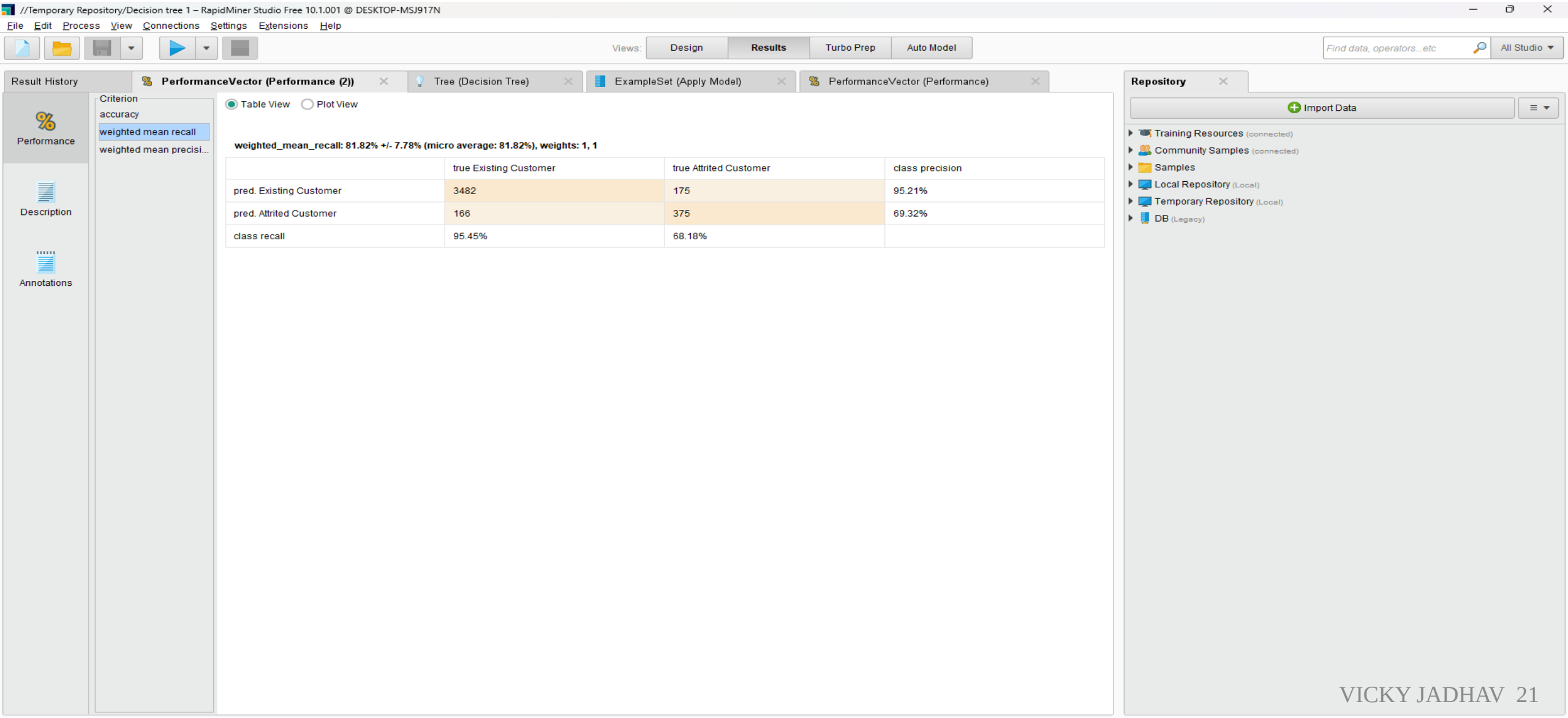




# CROSS VALIDATION

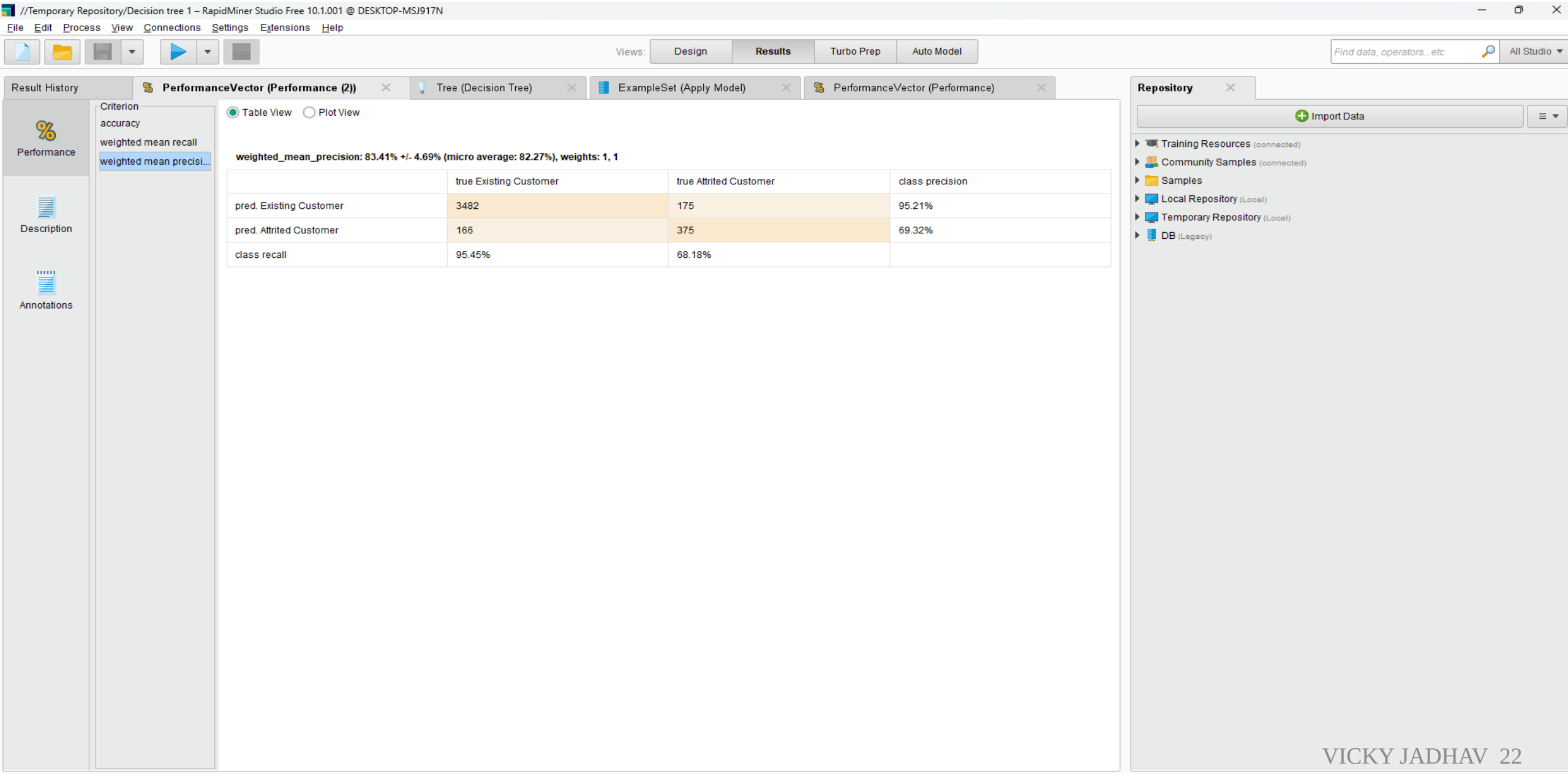
# ACCURACY


//Temporary Repository/Decision tree 1* – RapidMiner Studio Free 10.1.001 @ DESKTOP-MSJ917N
File Edit Process View Connections Settings Extensions Help
Views: Design Results Turbo Prep Auto Model
Find data, operators...etc
All Studio
Result History
PerformanceVector (Performance (2))
Tree (Decision Tree)
ExampleSet (Apply Model)
PerformanceVector (Performance)
Performance
Description
Annotations
Criterion
accuracy
weighted mean recall
weighted mean precisi...
Table View
Plot View
accuracy: 93.14% +/- 1.09% (micro average: 93.14%)
true Existing Customer
true Attrited Customer
class precision
pred. Existing Customer
3531
171
95.38%
pred. Attrited Customer
117
379
76.41%
class recall
96.79%
68.91%
Repository
Import Data
Training Resources (connected)
Community Samples (connected)
Samples
Local Repository (Local)
Temporary Repository (Local)
DB (Legacy)

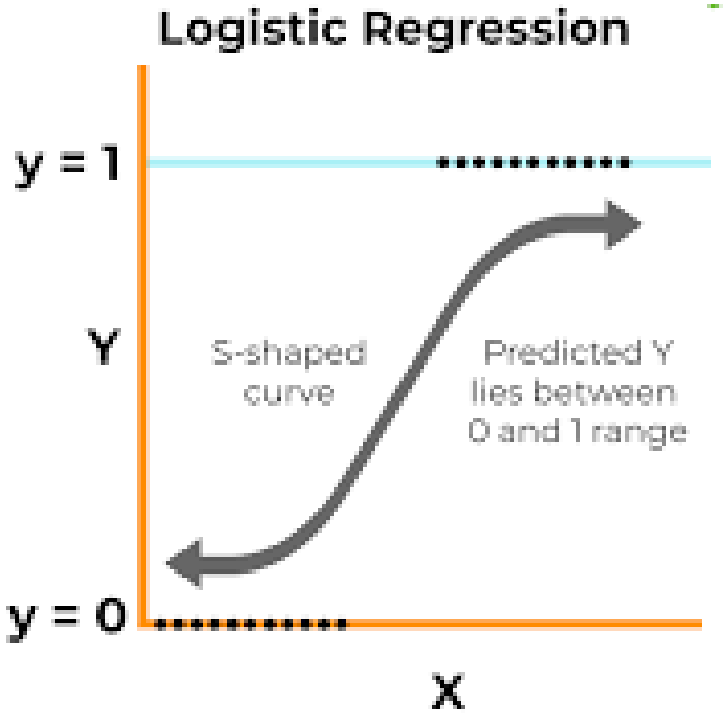
# MEAN RECALL


//Temporary Repository/Decision tree 1 – RapidMiner Studio Free 10.1.001 @ DESKTOP-MSJ917N
File Edit Process View Connections Settings Extensions Help
Views: Design Results Turbo Prep Auto Model
Find data, operators...etc
All Studio
Result History
PerformanceVector (Performance (2))
Tree (Decision Tree)
ExampleSet (Apply Model)
PerformanceVector (Performance)
Performance
Description
Annotations
Criterion
accuracy
weighted mean recall
weighted mean precisi...
Table View
Plot View
weighted_mean_recall: 81.82% +/- 7.78% (micro average: 81.82%), weights: 1, 1
true Existing Customer
true Attrited Customer
class precision
pred. Existing Customer
3482
175
95.21%
pred. Attrited Customer
166
375
69.32%
class recall
95.45%
68.18%
Repository
Import Data
Training Resources (connected)
Community Samples (connected)
Samples
Local Repository (Local)
Temporary Repository (Local)
DB (Legacy)

# MEAN PRECISION



- Introduction to Logistic Regression
- Dependent and Independent Variables
- Model Building and Evaluation
- Conclusion



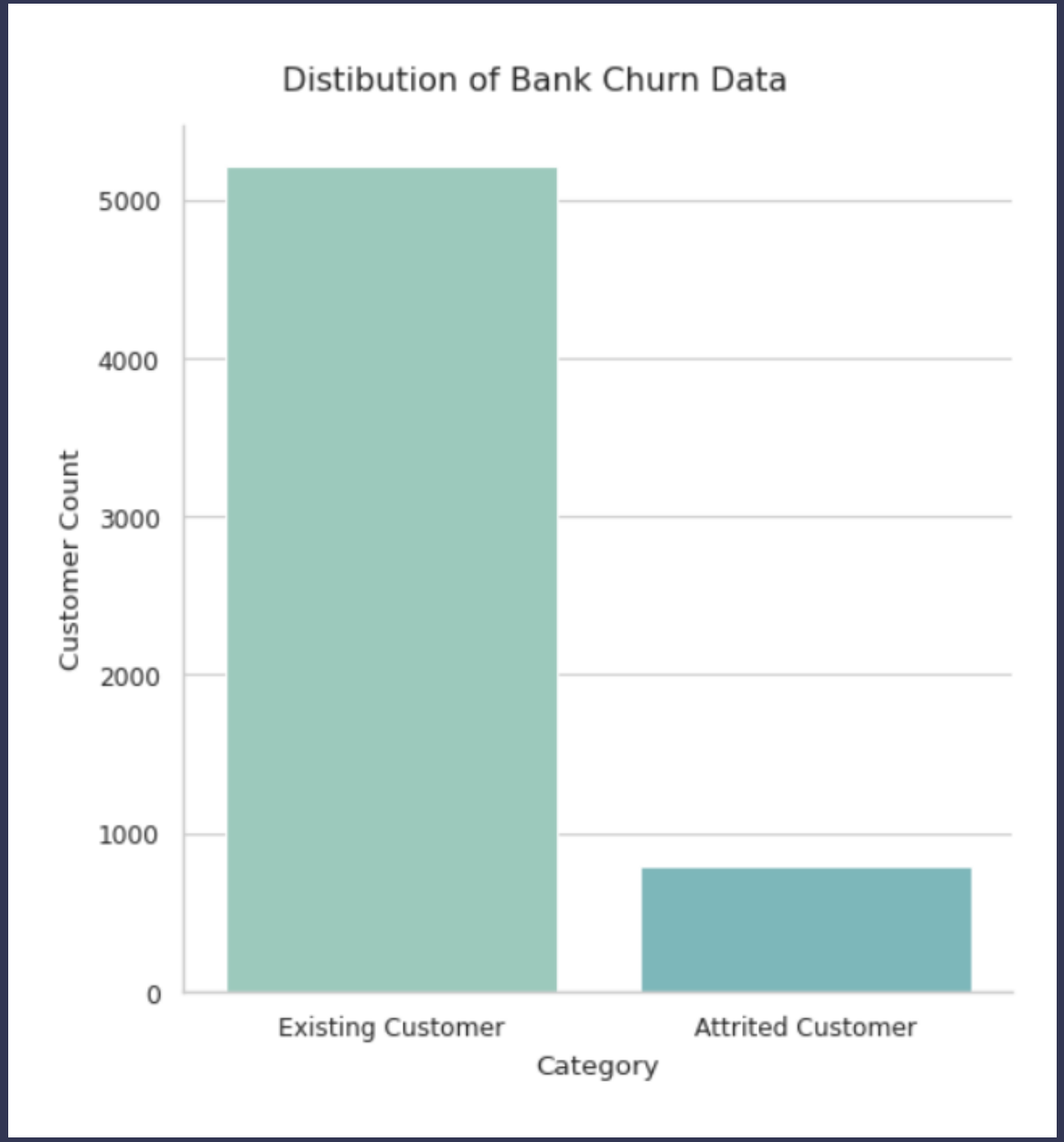

**Introduction to Logistic Regression**

- Logistic regression is a statistical method used to analyze the relationship between a dependent variable and one or more independent variables.
- The bank churners dataset is a popular dataset used to predict customer churn in banks. It contains information about customers who have either stayed with the bank or left the bank. The dataset includes variables such as age, income, credit score, and transaction history.

# Distribution of Dataset

```
sns.set_style('whitegrid')
sns.set_context('paper')
sns.set_palette('GnBu_d')
a = sns.catplot(x='Attrition Flag', data=data, kind='count')
a.fig.suptitle('Distibution of Bank Churn Data', y=1.03)
a.set(ylabel='Customer Count', xlabel='Category')
plt.show()
```

Overall,the label in the dataset has a majority of existing customers and very few attrited customers, so we can conclude that the dataset is imbalanced. In this case, the model trained on this dataset may be biased towards predicting existing customers and may have a low accuracy in predicting attrited customers.



## Dependent and Independent Variables

- In logistic regression, the dependent variable is Attrition Flag, which is categorized as Existing Customers and Attrited Customers. So, using the label encoder, the dependent variable is converted into numeric format, whether or not the customer has churned (0) or stayed (1).
- The independent variables are the predictor variables that are used to predict the probability of the dependent variable. In the bank churners dataset, the independent variables include 8 attributes, which were extracted in the first phase of the project.

Label Encoder Converting categorical values into numeric format

```
[13] # Gender: M = 1, F = 0
     # Attrition Flag = Existing Customer = 1, Attrited Customer = 0

     categorical_columns = ['Gender','Attrition Flag']

     # Apply label encoding to the categorical columns
     label_encoder = LabelEncoder()
     for col in categorical_columns:
         data[col] = label_encoder.fit_transform(data[col].astype(str))

     # Convert the columns to float data type
     data[categorical_columns] = data[categorical_columns].astype(float)
```

# Normalization

```
from sklearn.preprocessing import MinMaxScaler

# Normalize the encoded data using MinMaxScaler
scaler = MinMaxScaler()
normalized_labels = scaler.fit_transform(np.array(data[col]).reshape(-1, 1))

print(normalized_labels)
```

- Normalization is important in machine learning because it helps to standardize the range and scale of features or variables in a dataset. By normalizing the features, we can ensure that each feature contributes equally to the model and avoid bias towards any particular feature. This can lead to improved performance and more accurate predictions.

## Model Building and Evaluation

- To build a logistic regression model, we first split the dataset into training and testing sets. We then use the training set to build the model and the testing set to evaluate its performance.
- There are several metrics used to evaluate the performance of a logistic regression model. These include accuracy, precision, recall, and F1 score. The goal is to build a model that has high accuracy and precision, while minimizing false positives and false negatives.

Features

```
[15] x = data[['Gender','Total Relationship Count','Dependent count','Months Inactive 12 mon','Contacts Count 12 mon','Total Revolving Bal
```

Label

```
[16] y = data[['Attrition Flag']]
```

```
[17] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

spliting the data for training and testing. We are training 80% of data and testing 20% of data that's why test size is 0.2 (20%) random_state is used to get same output in every device

# Dummy Classifier

## Dummy Classifier

```
from sklearn.dummy import DummyClassifier

min_max_scaler = preprocessing.MinMaxScaler()
X_train = min_max_scaler.fit_transform(X_train)
X_test = min_max_scaler.transform(X_test)

# DummyClassifier to predict only target 0
dummy = DummyClassifier(strategy='most_frequent').fit(X_train, y_train)
dummy_pred = dummy.predict(X_test)

# checking unique labels
print('Unique predicted labels: ', (np.unique(dummy_pred)))

# checking accuracy
print('Test score: ', accuracy_score(y_test, dummy_pred))
```

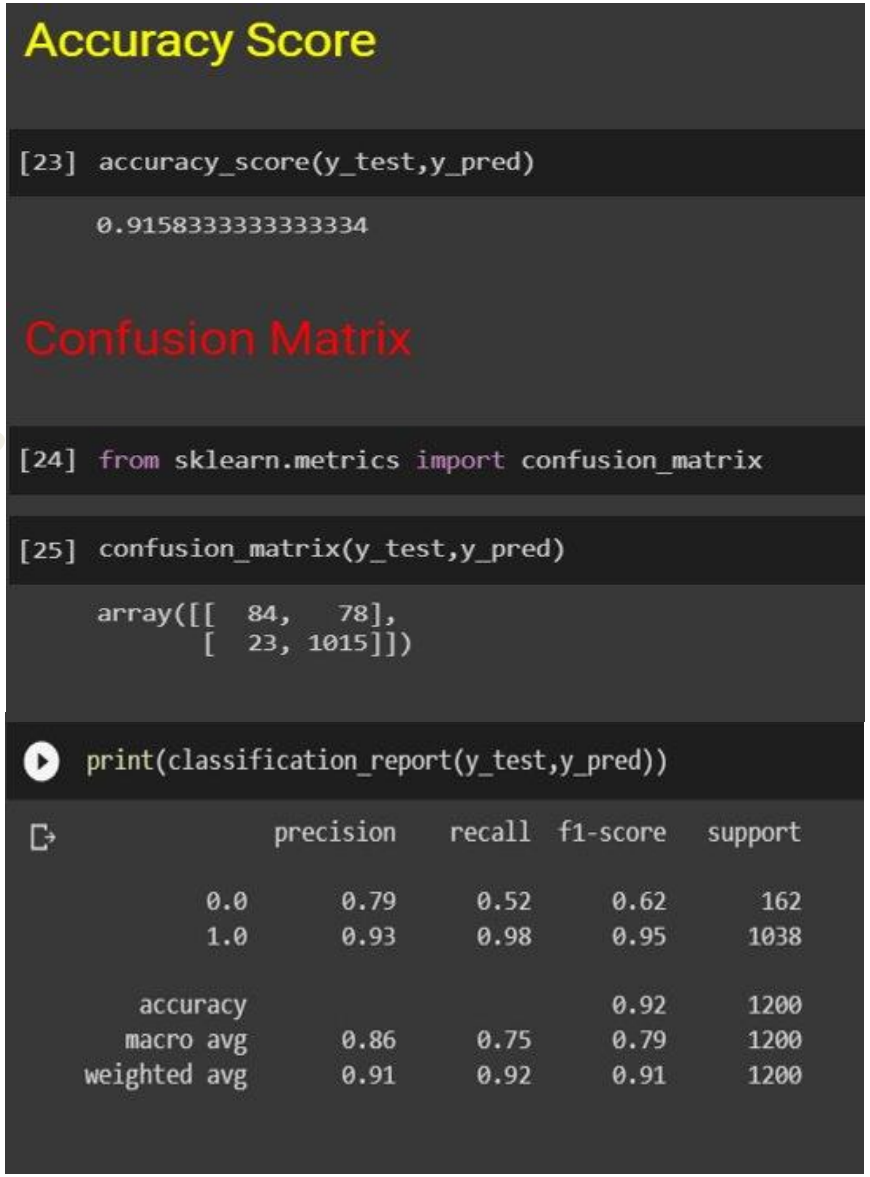
```
Unique predicted labels:  [1.]
Test score:  0.865
```

Logistic Regression

# Logistic Regression

```
[19] from sklearn.linear_model import LogisticRegression
     classifier = LogisticRegression()

[20] classifier.fit(X_train,y_train)

     /usr/local/lib/python3.9/dist-packages/sklearn/utils/va
       y = column_or_1d(y, warn=True)
     ▾ LogisticRegression
     LogisticRegression()

[21] y_pred = classifier.predict(X_test)

[22] y_pred

     array([1., 1., 1., ..., 1., 1., 1.])
```

- **Accuracy score:** Accuracy score is a metric that measures the proportion of correct predictions made by the model. It is calculated as the ratio of the number of correct predictions to the total number of predictions made by the model.
- **Confusion matrix:** A confusion matrix is a table that shows the number of true positive, false positive, true negative, and false negative predictions made by the model. It is often used to evaluate the performance of a binary classification model.
- **Confusion report:** A confusion report is a visual representation of the confusion matrix. It provides a more detailed view of the model's performance by showing the precision, recall, F1 score, and support for each class.




**Accuracy: 92%**
**Precision: 93%**
**Recall: 98%**

# USING RAPID MINER

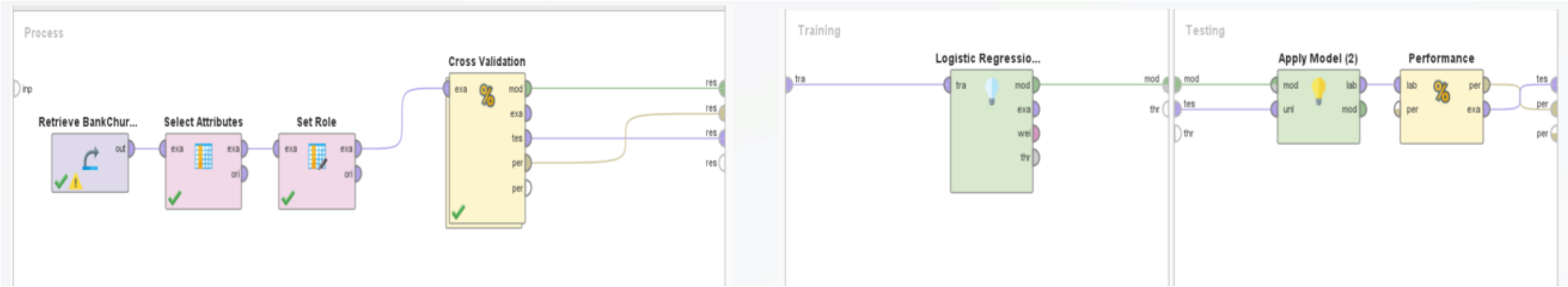



Table View  Plot View

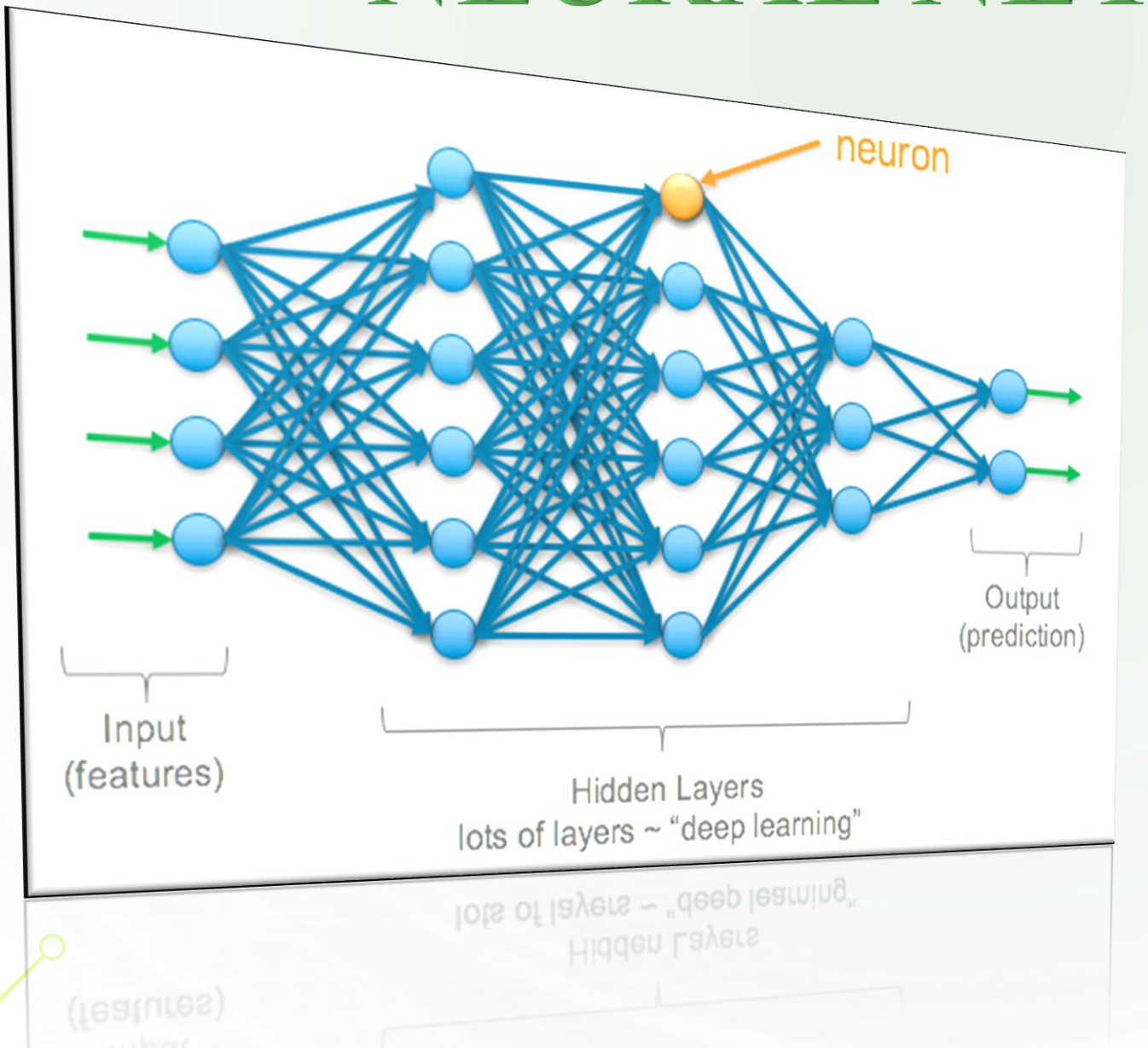
**accuracy: 92.20% +/- 26.82% (micro average: 92.20%)**

| | true Existing Customer | true Attrited Customer | class precision |
|---|---|---|---|
| pred. Existing Customer | 5068 | 324 | 93.99% |
| pred. Attrited Customer | 144 | 462 | 76.24% |
| class recall | 97.24% | 58.78% | |

# Conclusion

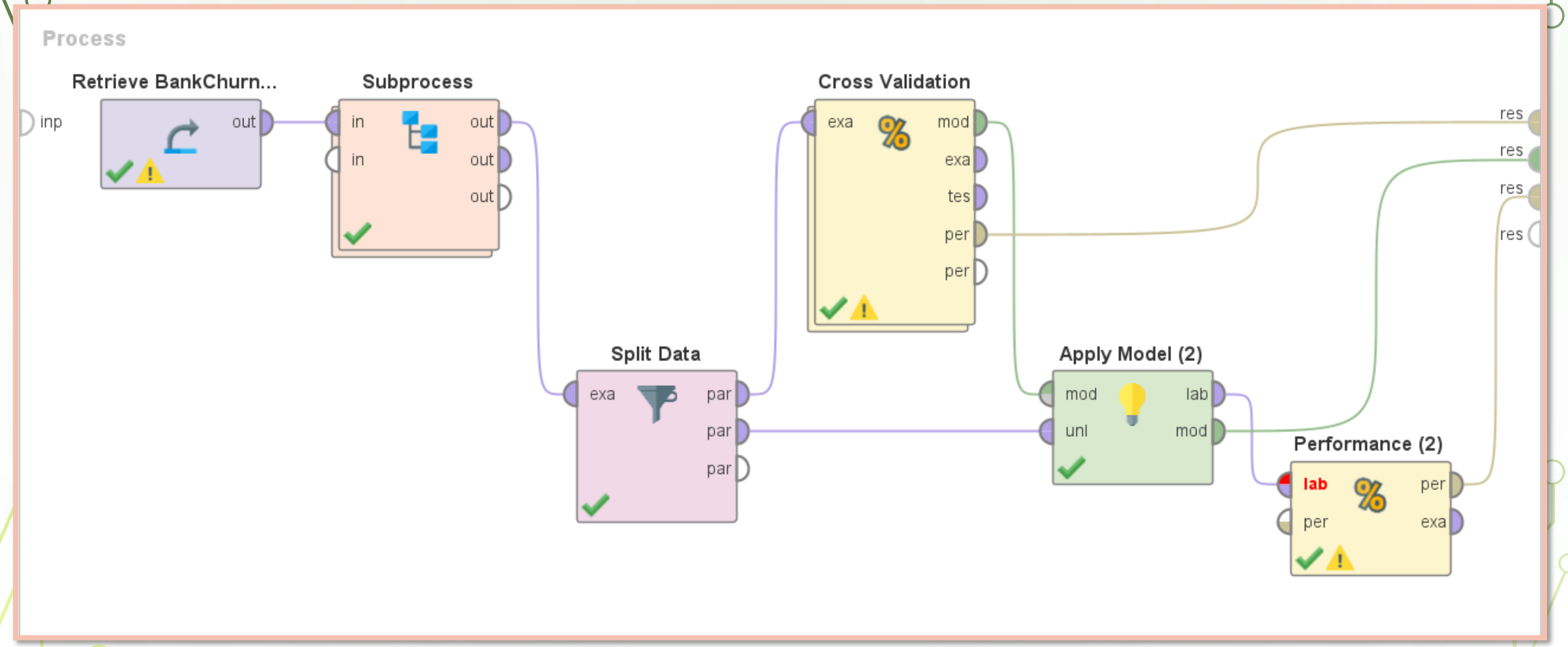
- Logistic regression is a powerful tool for predicting binary outcomes in a variety of industries. By understanding the relationships between independent and dependent variables, we can build accurate models.
- The bank churners dataset is a valuable resource for predicting customer churn in banks. By using logistic regression, we can identify the most important factors that contribute to churn and develop strategies to retain customers.

# NEURAL NETWORK



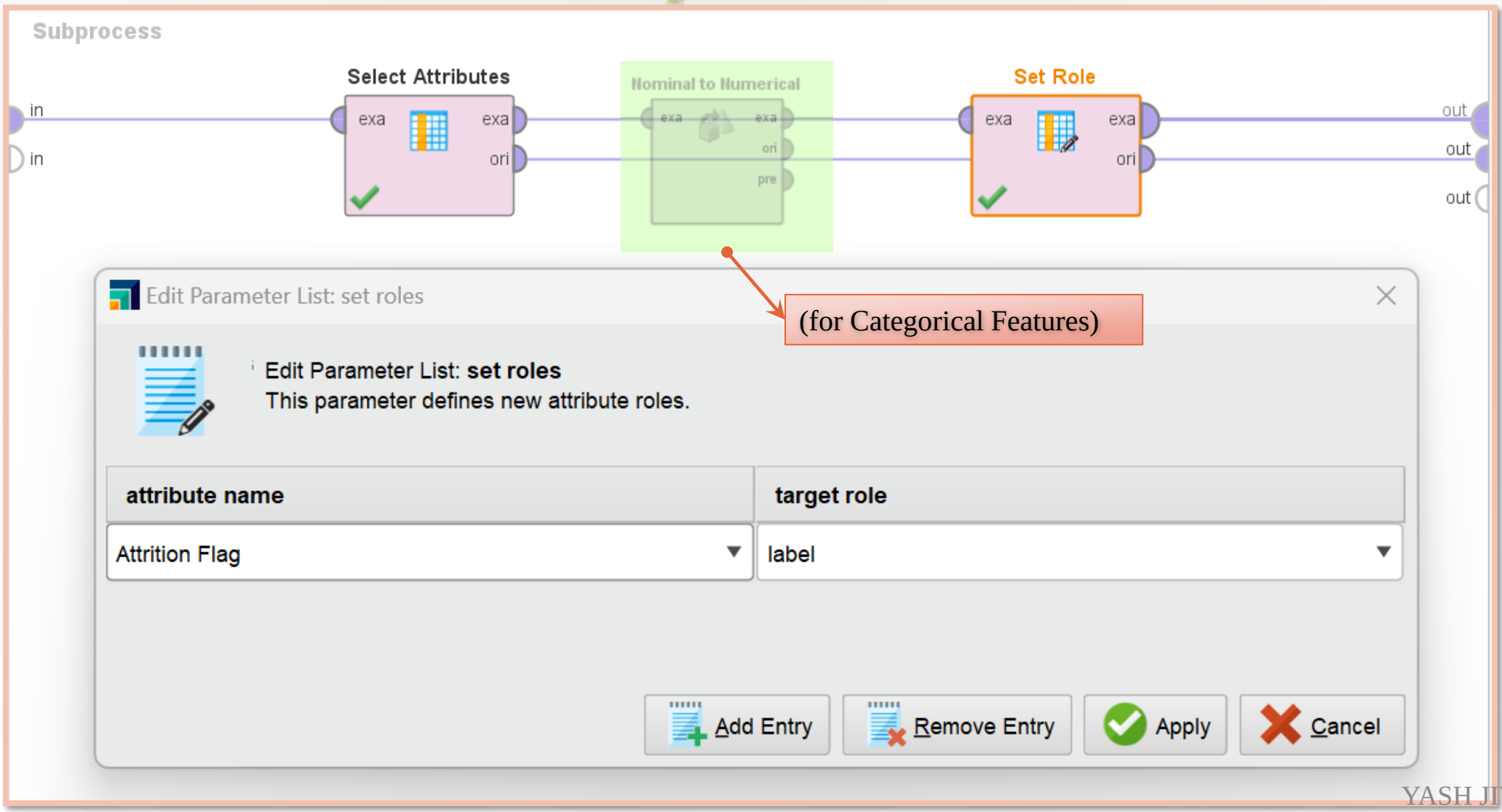

- Neural Network is a ML algorithm which is comprised of 3 layers of nodes, containing an input layer, one or more hidden layers, and an output layer.
- Each node, or artificial neuron, connects to another and has an associated weight and threshold
- Neural networks rely on training data to learn and improve their accuracy over time.

# RapidMiner Model Layout



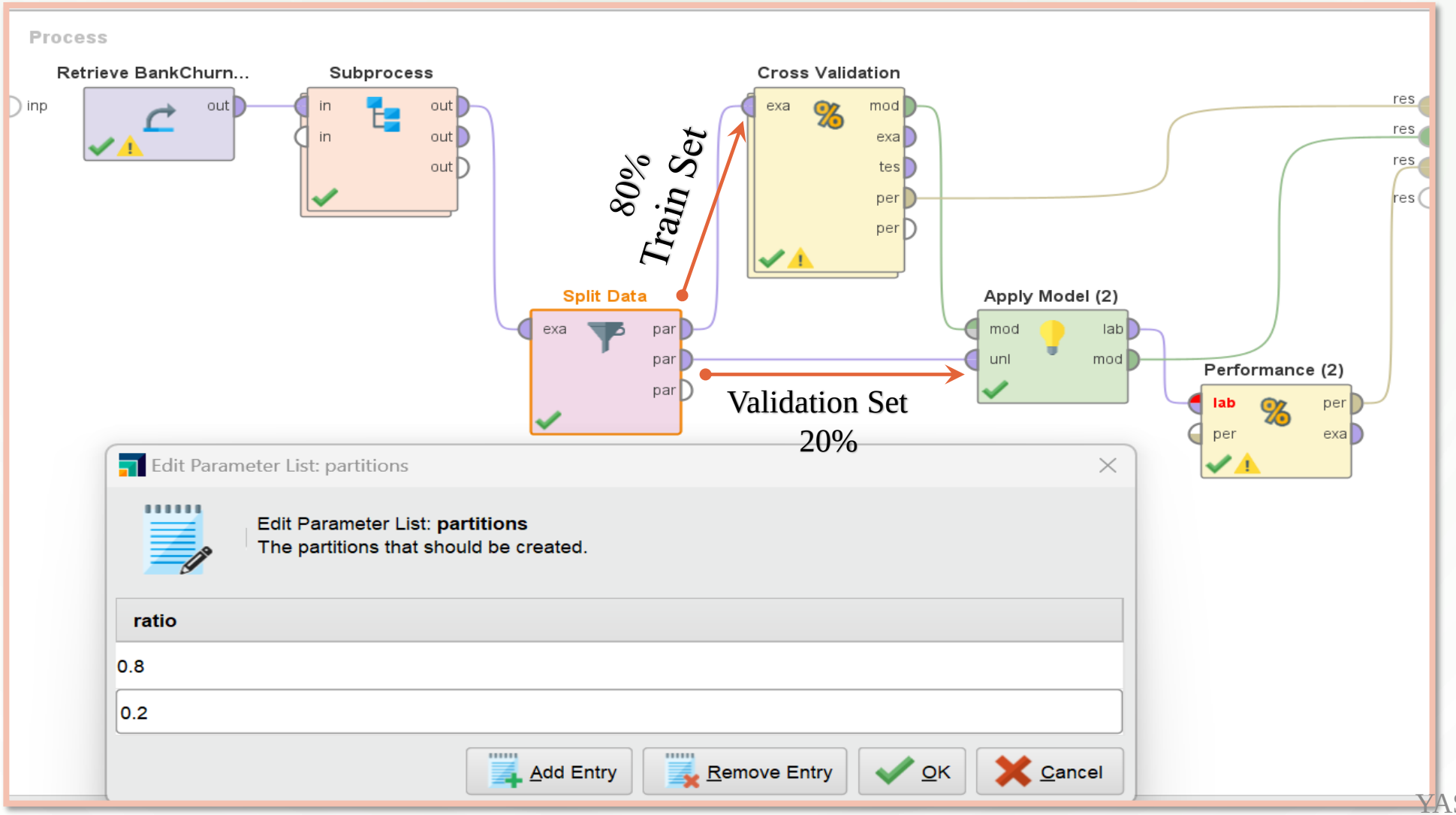
# RapidMiner Model Process

## Subprocess



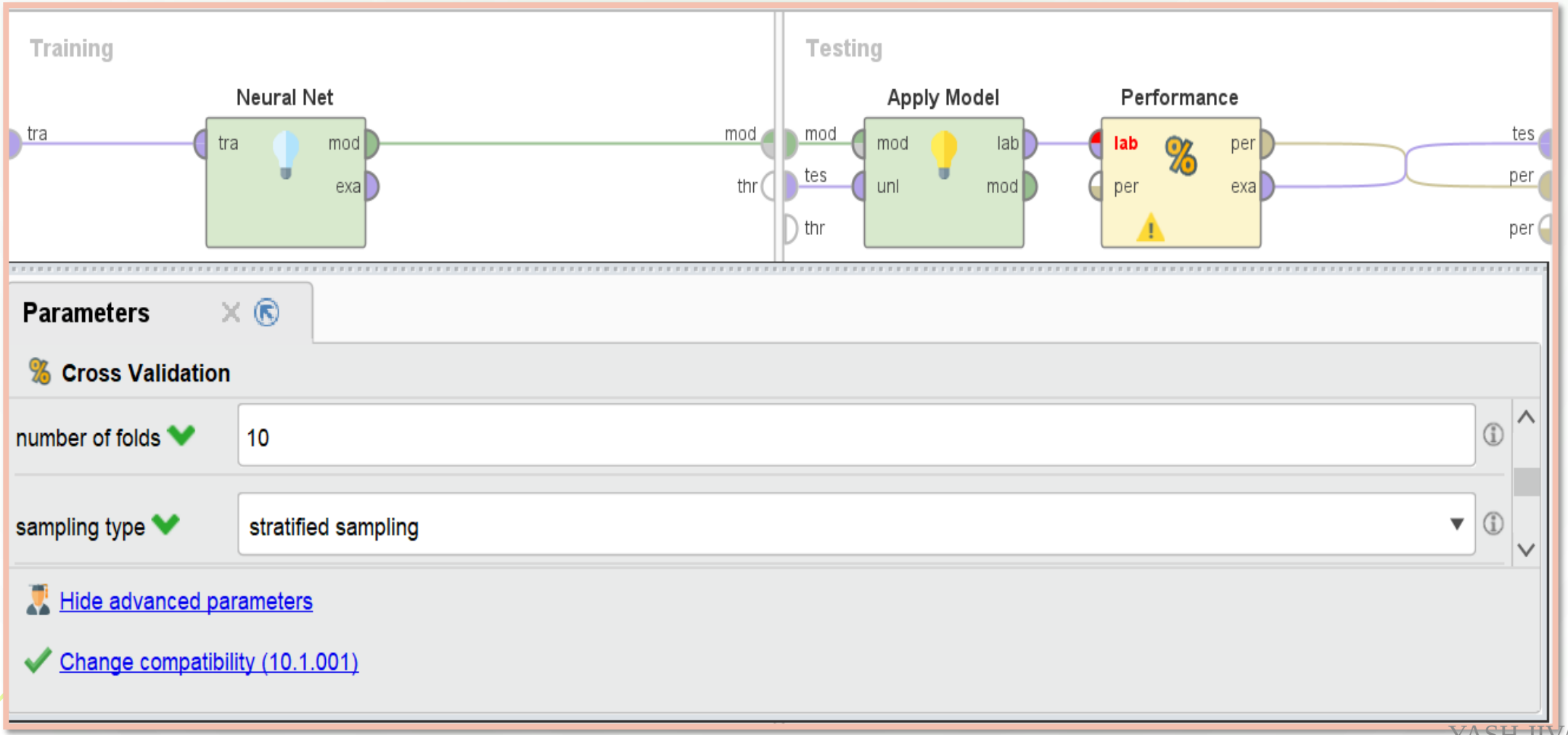
# RapidMiner Model Process

## Split - Data



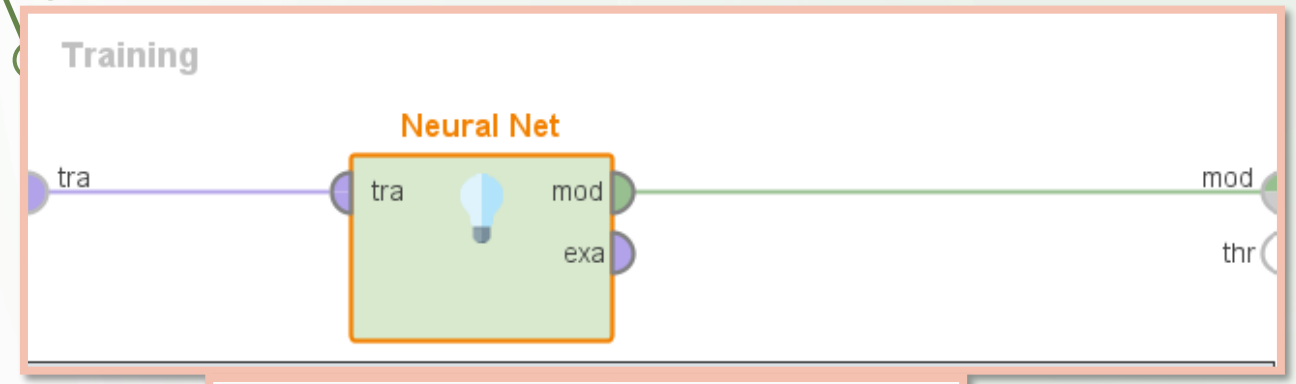
# RapidMiner Model Process

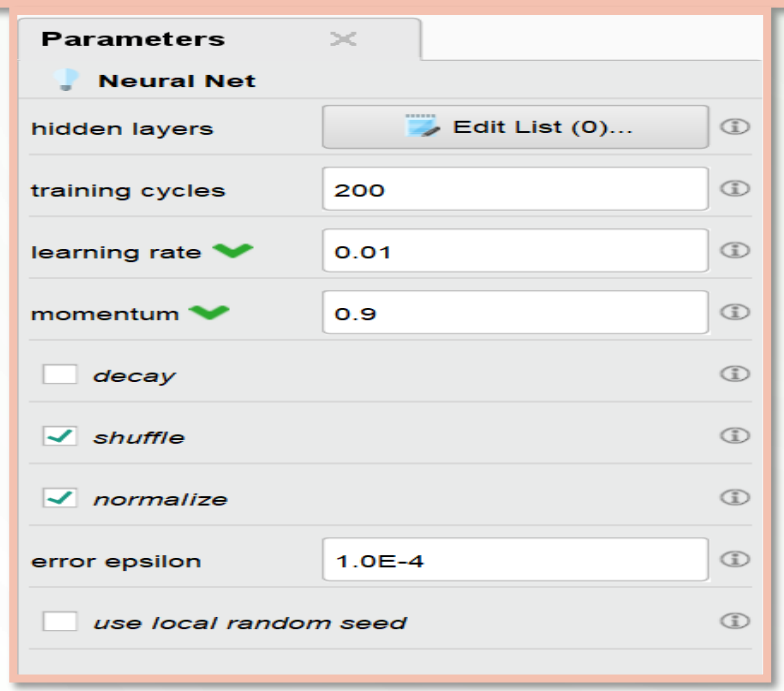
## Cross Validation



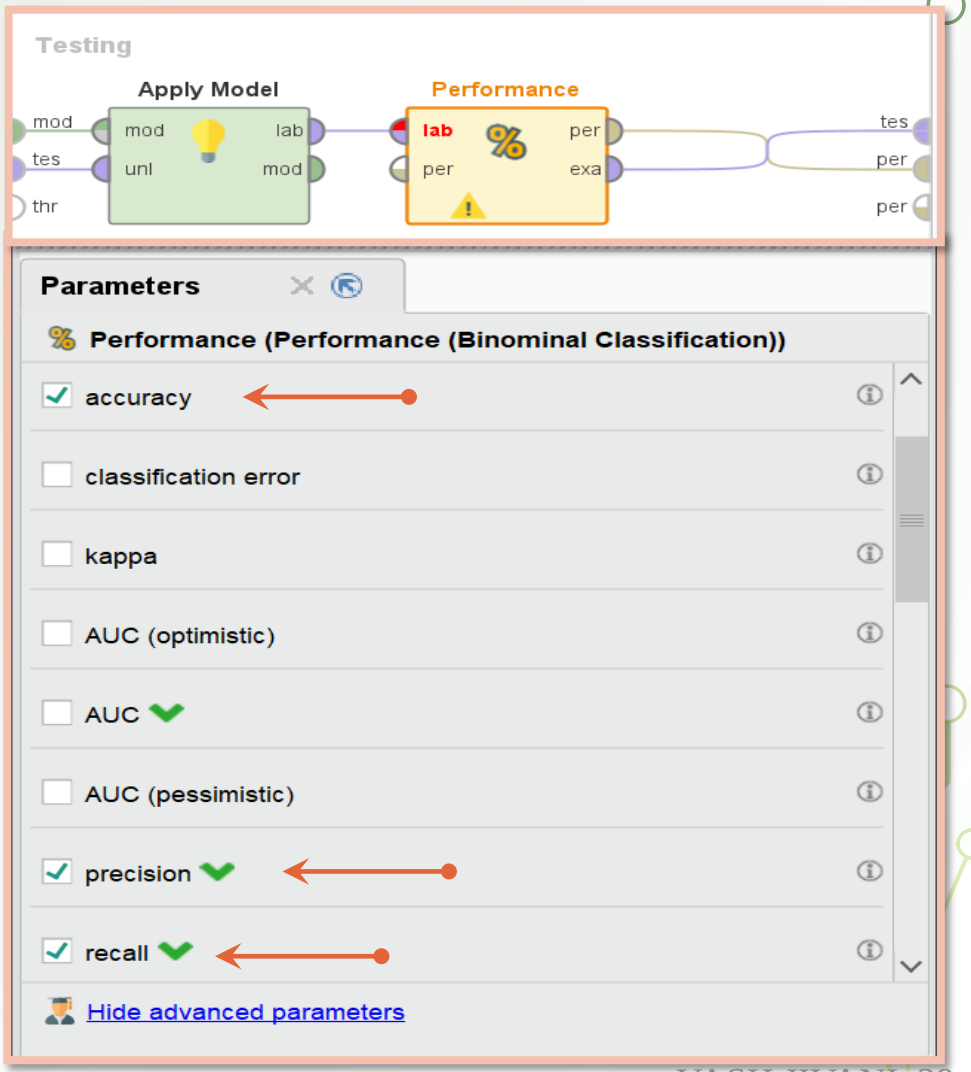
# RapidMiner Model Process

## Model Parameter



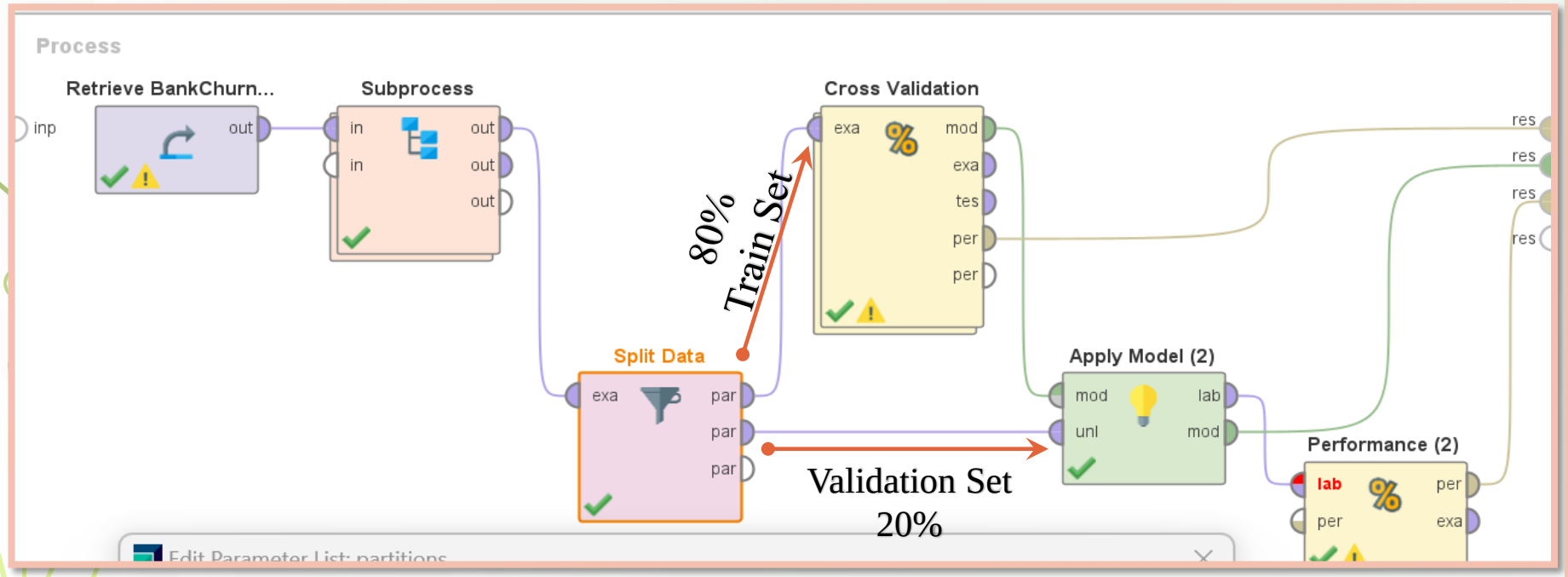



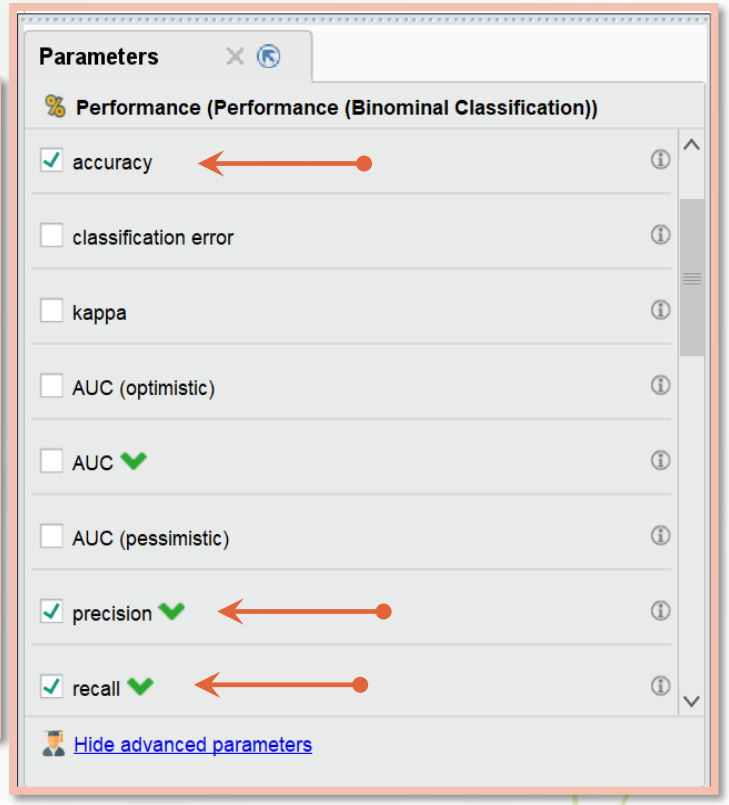

## Performance Parameter



# RapidMiner Model Process

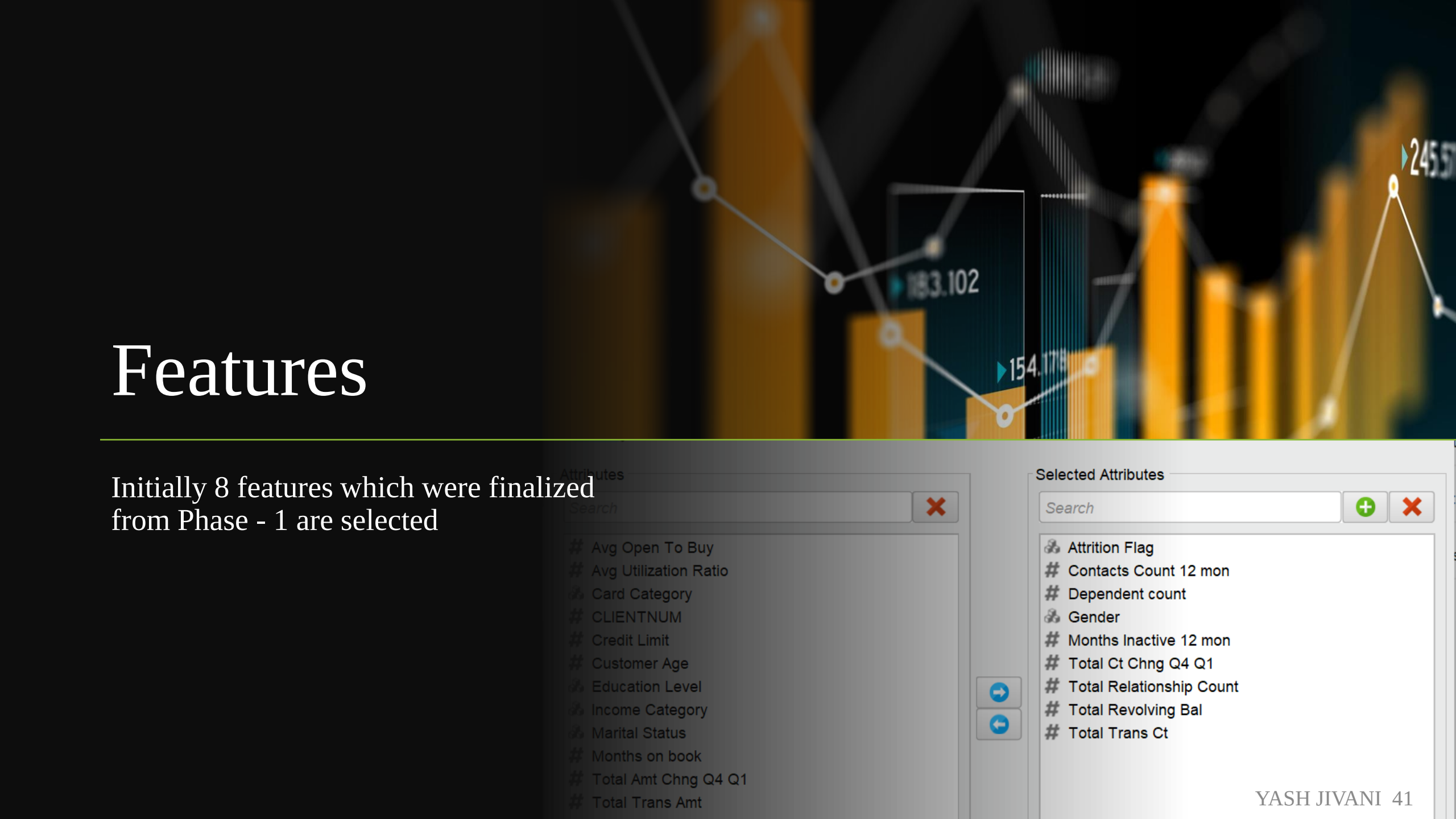
## Validation Set Performance



# Features

Initially 8 features which were finalized from Phase - 1 are selected



# MODEL RESULTS – CONFUSION MATRIX

## Performance Score – Training Set

**accuracy: 93.98%** **+/- 1.04% (micro average: 93.98%)**

| | true Existing Customer | true Attrited Customer | class precision |
|---|---|---|---|
| pred. Existing Customer | 4075 | 194 | 95.46% |
| pred. Attrited Customer | 95 | 435 | 82.08% |
| class recall | 97.72% | 69.16% | |

Accuracy – 93.98%
Mean Precision – 88.77%
Mean Recall – 83.44%

## Performance Score – Validation Set

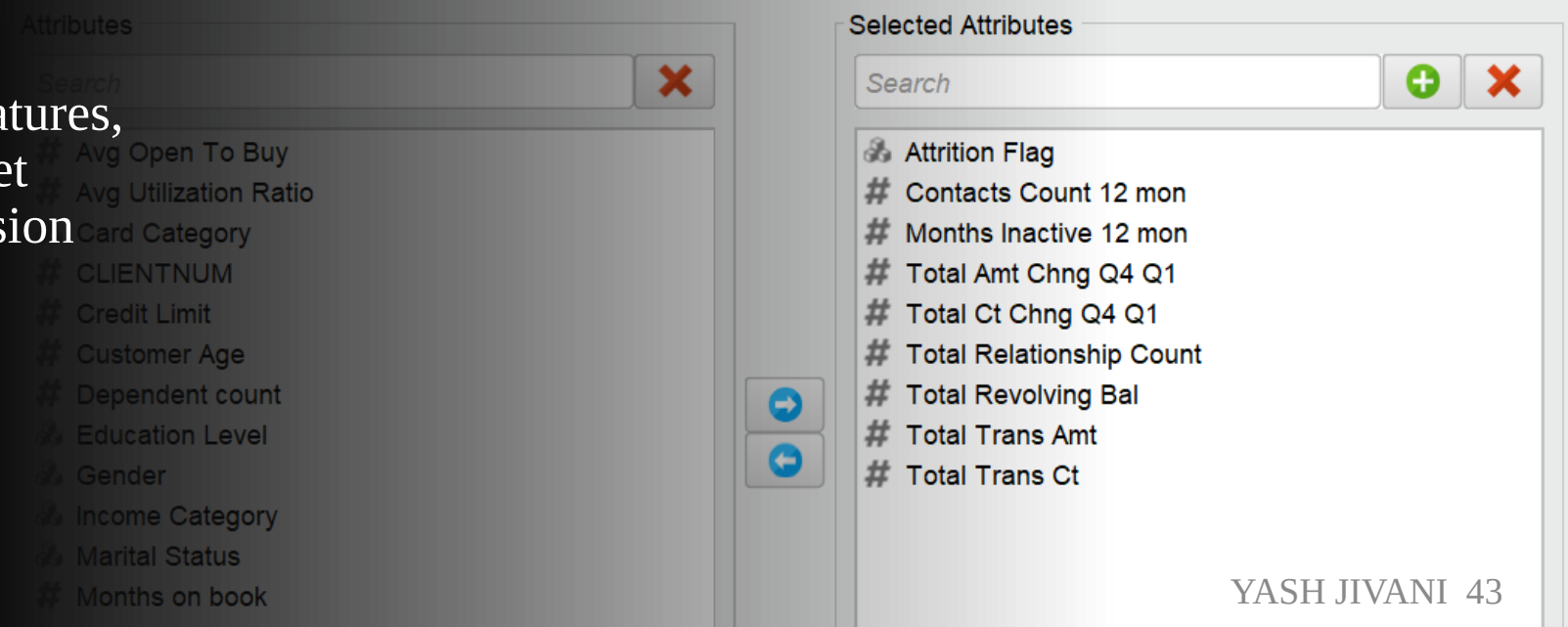
**accuracy: 92.83%**

| | true Existing Customer | true Attrited Customer | class precision |
|---|---|---|---|
| pred. Existing Customer | 1011 | 55 | 94.84% |
| pred. Attrited Customer | 31 | 102 | 76.69% |
| class recall | 97.02% | 64.97% | |

Accuracy – 92.83%
Mean Precision – 85.77%
Mean Recall – 80.99%

# Features for Neural Net

Finally, after multiple iterations of selection & removal of different features, attributes which gave the Neural Net model with highest accuracy, precision & recall are selected.

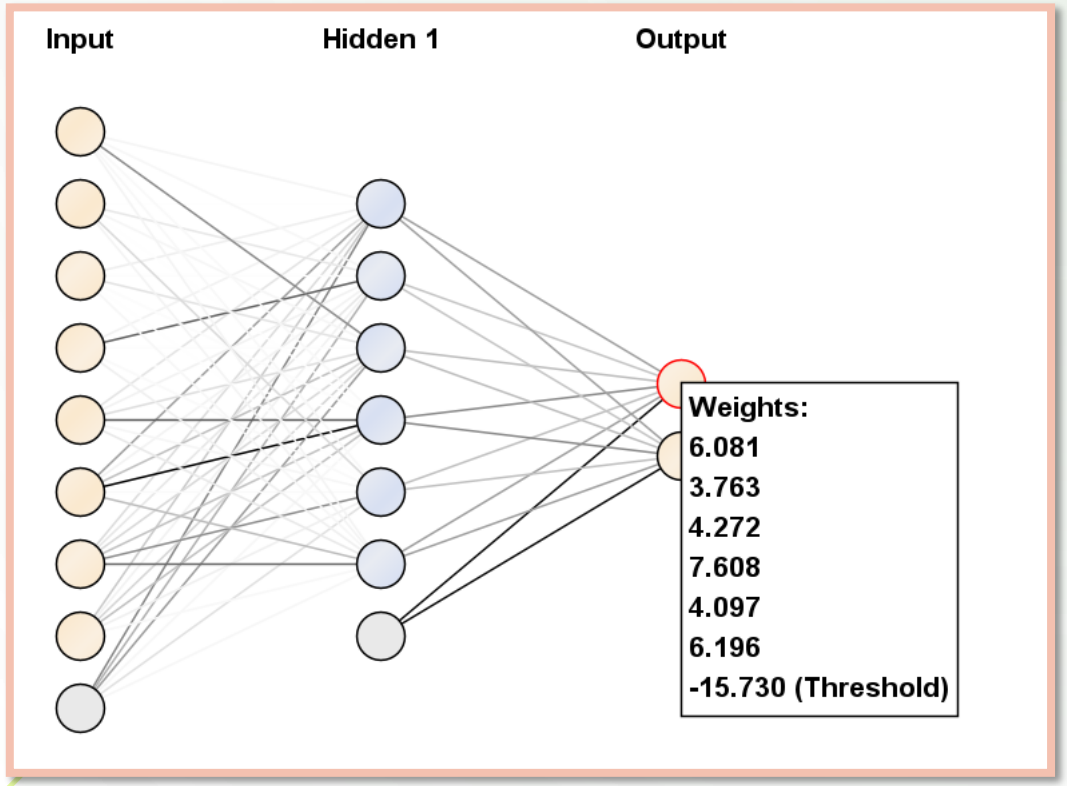



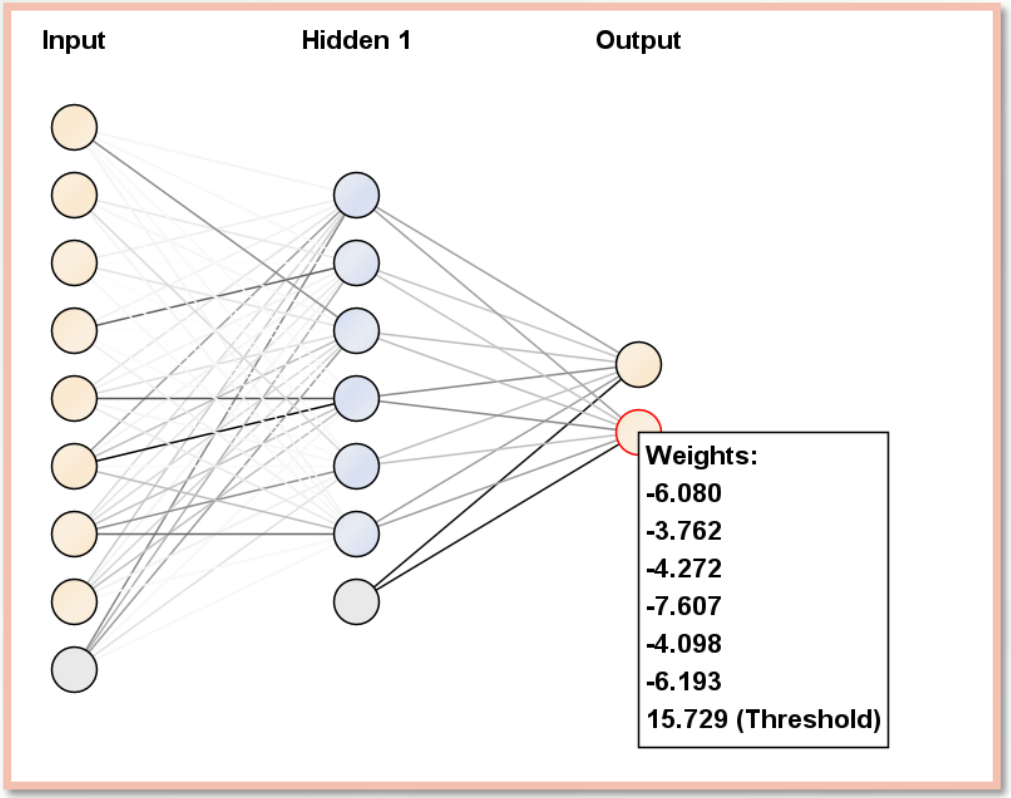
# Neural Net Model

Input Layer– 8 + 1 Nodes

Hidden Layer – 6 + 1 Nodes

Output Layer – 2 Nodes



# MODEL RESULTS – CONFUSION MATRIX

## Performance Score - Training Set

**accuracy: 96.08% +/- 0.78% (micro average: 96.08%)**

| | true Existing Customer | true Attrited Customer | class precision |
|---|---|---|---|
| pred. Existing Customer | 4089 | 107 | 97.45% |
| pred. Attrited Customer | 81 | 522 | 86.57% |
| class recall | 98.06% | 82.99% | |

Accuracy – 96.08%
Mean Precision – 92.01%
Mean Recall – 90.53%

## Performance Score – Validation Set

**accuracy: 96.16%**

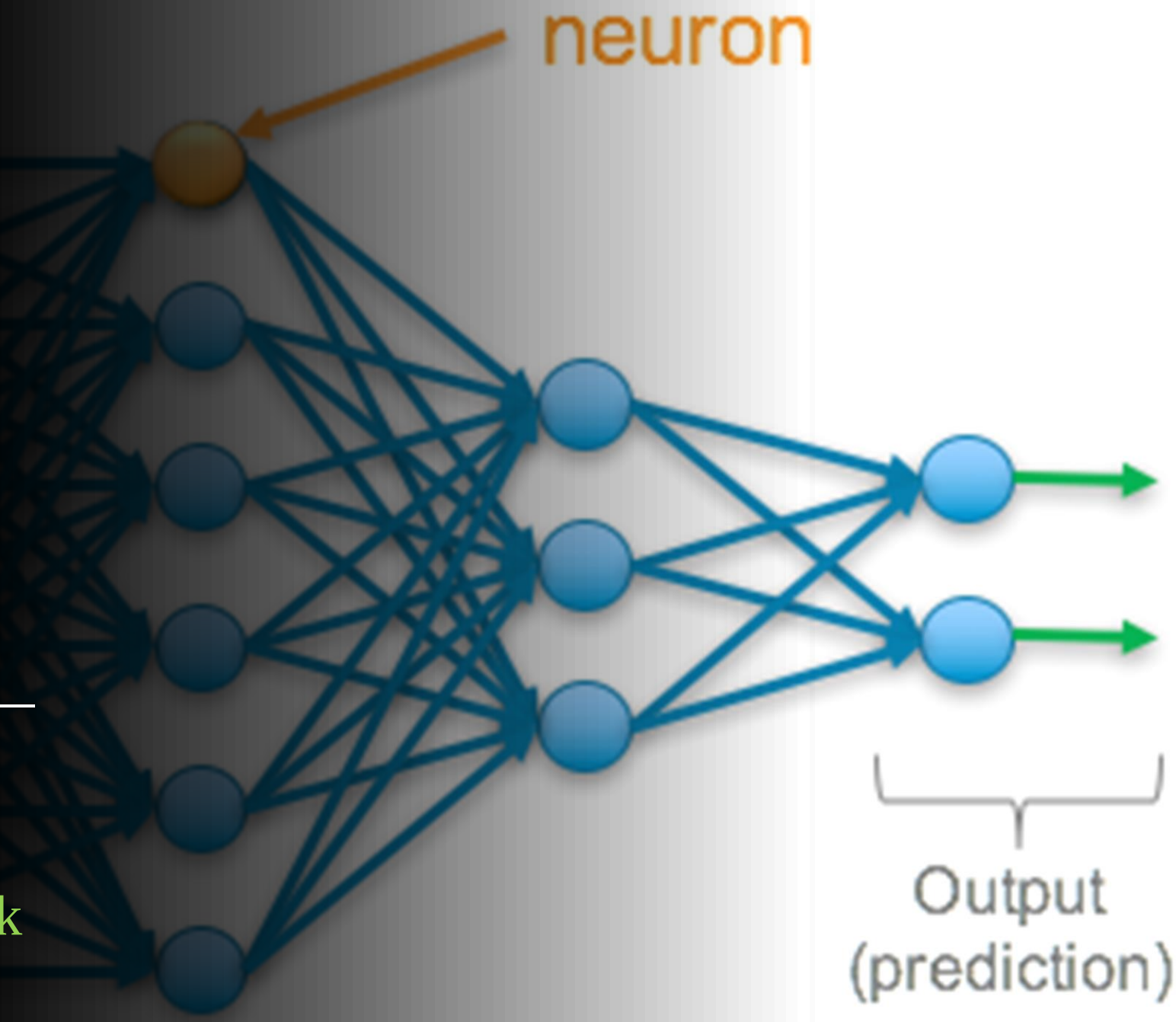
| | true Existing Customer | true Attrited Customer | class precision |
|---|---|---|---|
| pred. Existing Customer | 1027 | 31 | 97.07% |
| pred. Attrited Customer | 15 | 126 | 89.36% |
| class recall | 98.56% | 80.25% | |

Accuracy – 96.16%
Mean Precision – 93.22%
Mean Recall – 89.41%

- SVM
- Decision Tree
- Logistic Regression
- Neural Network

# Model Selection

Based on overall performance & score of Accuracy, Neural Network Model is selected



# THANK YOU!

## Any Questions