HW1_Part B

Group members: Dola Qiu, Seth Abayo, Lanston Chen, Yizhou Sun, Yihua Wang

30 Points (Python):

a) (5 points) Build a decision tree model that predicts whether a consumer will terminate his/her contract. In particular, I would like for you to create a decision tree using entropy with no max depth. Some possible issues / hints to think about: using training vs. test datasets.

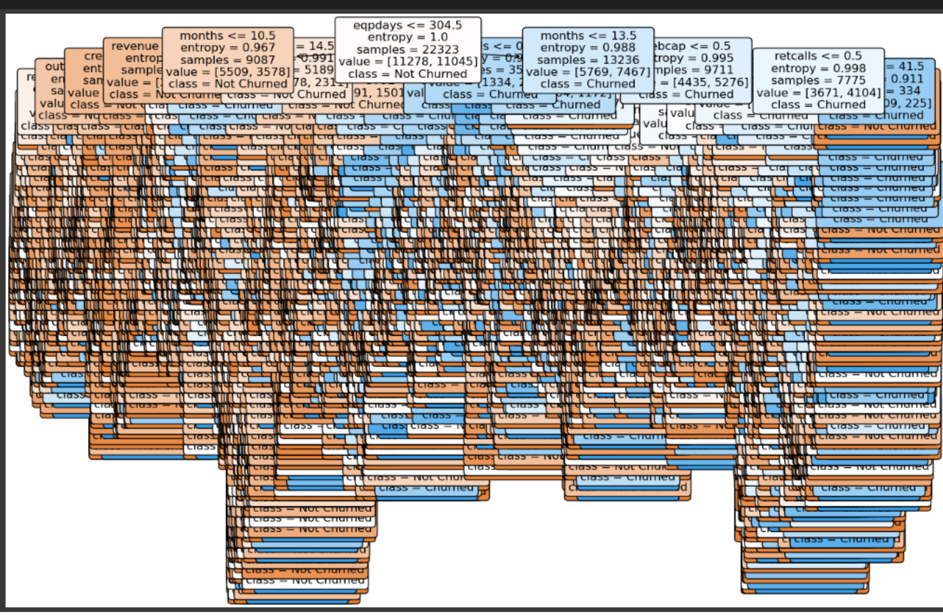
1. Decision Tree Model Building

- Split the data into training and testing sets.
- Built an initial Decision Tree model using the 'entropy' criterion and without max depth specified.

5. Creating Decision tree instance | Fiitng the model and Predicting

```
clf = DecisionTreeClassifier(criterion='entropy') # Decision tree classifier instance using entropy and with no max depth specified

# Fit the data
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)
```

```
# Plot the tree
plt.figure(figsize=(15,10))

tree.plot_tree(clf.fit(X_train,y_train),
               feature_names=X.columns,
               filled=True,
               class_names=class_names,
               rounded=True,
               fontsize=12)
plt.show()
```



b) (5 points) Explore how well the decision trees perform for several different parameter values (e.g., for different splitting criteria).

First we create a new dictionary variable called “params”, which contains all different values of criterion\ max_depth and min_samples_leaf.

```
params = {
    'criterion': ['gini', 'entropy'],
    'max_depth' : range(1,20),
    'min_samples_leaf': range(1,20),

}
```

And then, we build a decision tree model using a 3-level nested For-loop traversing each parameter.

```
for criterion in params['criterion']:
    for max_depth in params['max_depth']:
        for min_samples_leaf in params['min_samples_leaf']:
            # Initialize Decision tree with different parameters
            clf = DecisionTreeClassifier(
                criterion=criterion,
                max_depth=max_depth,
                min_samples_leaf= min_samples_leaf
            )
```

c) (5 points) Discuss the model (decision tree) that provides the best predictive performance from experimenting with different parameter values in question (b).

In our for_loop function, we choose the model with the highest accuracy value.

```
Best accuracy is: 0.6047240802675585
Best parameters : {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1}
```

Compared to the initial model, the new one increases both precision and recall of the class1.

- Precision is more important if the business goal is to reduce the cost of misclassifying churned users (e.g., don't want to give discounts or offers to users who won't actually churn).
- If the business goal is to ensure that as many churned users as possible are identified so that steps can be taken to retain them, then recall may be more important. In most of user Churn cases, the objective is to find more possible churn user, so looking recall are more reasonable to estimate the Chuen predict ML model

d) (15 points) Present a brief overview of your predictive modeling process, explorations, and discuss your results. That is, you need to lay out the steps you have taken in order to build and evaluate the decision tree model. For instance, how did you explore the data set before you built the model? Write this report in a way that the upper level management of the team would understand what you are doing. Why is the decision tree an appropriate model for this problem? How can we evaluate the predictive ability of the decision tree? If you build decision trees with different splitting criteria, which decision tree would you prefer to use in practice? Make sure you present information about the model “goodness” (please report the confusion matrix, predictive accuracy, classification error, precision, recall, f-measure).

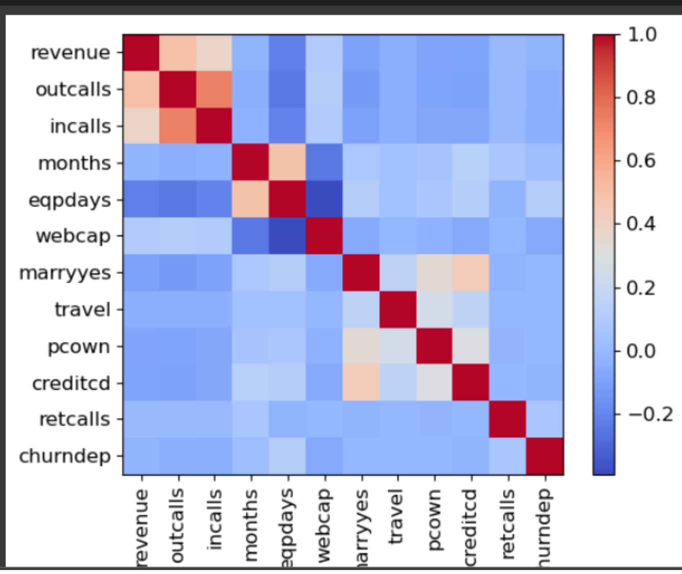
Predictive Modeling Process

1. Data Exploration

- Used the methods of ‘head’ and ‘shape’ to have a brief understanding of what the data look like.
- Used a heatmap to understand the correlation between different variables.
- Checked for negative values in 'revenue' and 'eqpdays' and replaced them with zero. (Since the rows with negative values in ‘eqpdays’ and ‘revenue’ can also provide other additional information for our prediction for the churn, we choose to change the negative values into 0 rather than just delete the whole rows.)
- Checked for class imbalance in the target variable 'churndep' and found it to be balanced.

Plotting correlation in heatmap

```
# Create a heatmap
plt.imshow(data.corr(), cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.xticks(range(len(data.corr().columns)), data.corr().columns, rotation=90)
plt.yticks(range(len(data.corr().columns)), data.corr().columns)
plt.show()
```

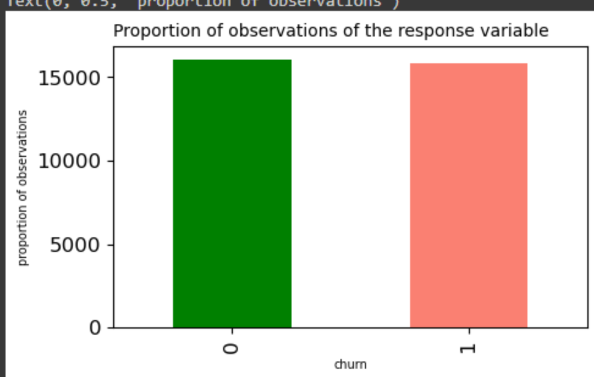


```
[ ] # Count how many neg values are in revenues and eqpdays
    data[(data['revenue'] < 0) ].shape, data[(data['eqpdays'] < 0) ].shape

    ((1, 12), (46, 12))
```

Check class imbalance

```
fig = plt.figure(figsize=(5,3))
ax = fig.add_subplot(111)
data['churndep'].value_counts().plot(kind='bar',
                                     ax=ax,
                                     color=['green','salmon'])

# set title and labels
ax.set_title('Proportion of observations of the response variable',
             fontsize=10, loc='left')
ax.set_xlabel('churn',
              fontsize=7)
ax.set_ylabel('proportion of observations',
              fontsize=7)
```

```
Text(0, 0.5, 'proportion of observations')
```



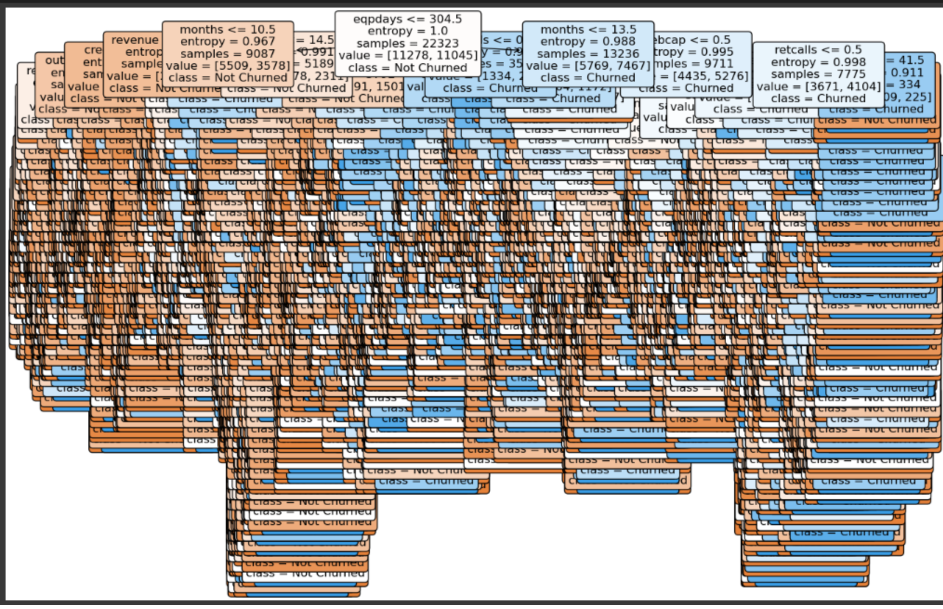
2. Model Building

- Split the data into training and testing sets.
- Built an initial Decision Tree model using the 'entropy' criterion and without max depth specified.

5. Creating Decision tree instance | Fiitng the model and Predicting

```
clf = DecisionTreeClassifier(criterion='entropy') # Decision tree classifier instance using entropy and with no max depth specified

# Fit the data
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)
```

```
[ ] # Plot the tree
    plt.figure(figsize=(15,10))

    tree.plot_tree(clf.fit(X_train,y_train),
                  feature_names=X.columns,
                  filled=True,
                  class_names=class_names,
                  rounded=True,
                  fontsize=12)
    plt.show()
```



3. Model Evaluation

- The initial model had an accuracy of 53.4%.

6. Evaluating model performance, visualizing confusion matrix

```
[ ] print("Accuracy: ", accuracy_score(y_test, y_pred))
    print(f'F1 Score: {f1_score(y_test, y_pred)}')
    print(f'Precision: {precision_score(y_test, y_pred)}')
    print(f'Recall: {recall_score(y_test, y_pred)}')

    Accuracy:  0.5338628762541806
    F1 Score: 0.526338147833475
    Precision: 0.5379939209726444
    Recall: 0.5151767151767151

[ ] print(f"\t\t The classification report  \n\n {classification_report(y_test, y_pred)}")

                  The classification report

                precision    recall  f1-score   support

           0       0.53      0.55      0.54      4758
           1       0.54      0.52      0.53      4810

    accuracy                           0.53      9568
   macro avg       0.53      0.53      0.53      9568
weighted avg       0.53      0.53      0.53      9568
```

4. Hyperparameter Tuning

- Experimented with different parameters like 'criterion', 'max_depth', and 'min_samples_leaf'.
- Systematically searched through various hyperparameter combinations to find the best-performing model configuration and improve model performance
- Found the best parameters that improved the model's accuracy to 60.4%.

```python
# define parameters dict and iterate thru all parameters while keeping the best ones

# I have refrenced to scikit-learn to define some of the below paramaters: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
params = {
    'criterion': ['gini', 'entropy'],
    'max_depth' : range(1,20),
    'min_samples_leaf': range(1,20),

}

best_params = {} # a dict to help us keep best parameters
best_accuracy = 0.0 # Initialize this to 0 and we keep update once we get new best accuracy
cnf_matrix_tuned = []
report = None

# Iterate thru all possible combination of params

for criterion in params['criterion']:
    for max_depth in params['max_depth']:
        for min_samples_leaf in params['min_samples_leaf']:
                # Initialize Decision tree with different parameters
            clf = DecisionTreeClassifier(
                criterion=criterion,
                max_depth=max_depth,
                min_samples_leaf= min_samples_leaf
            )

            clf.fit(X_train,y_train)

            #Make predictions
            y_pred = clf.predict(X_test)
            accuracy = accuracy_score(y_test, y_pred)

            #update best_accuracy variable if current accuracy is greater than best_accuracy
            if accuracy > best_accuracy:
                best_accuracy = accuracy

                # Also add best parameters associated with best accuracy
                best_params = {
                    'criterion': criterion,
                    'max_depth': max_depth,
                    'min_samples_leaf': min_samples_leaf
```

5. Best Model

Choose the model with the highest accuracy. (We chose to limit the max-depth under 20 because we do not want the final model to have too much nodes, which can cause overfitting)

- Criteria: entropy
- Max Depth: 5
- Min Samples Leaf: 1
- Best Accuracy: 60.4%

```
Best accuracy is: 0.6047240802675585
Best parameters : {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1}
```

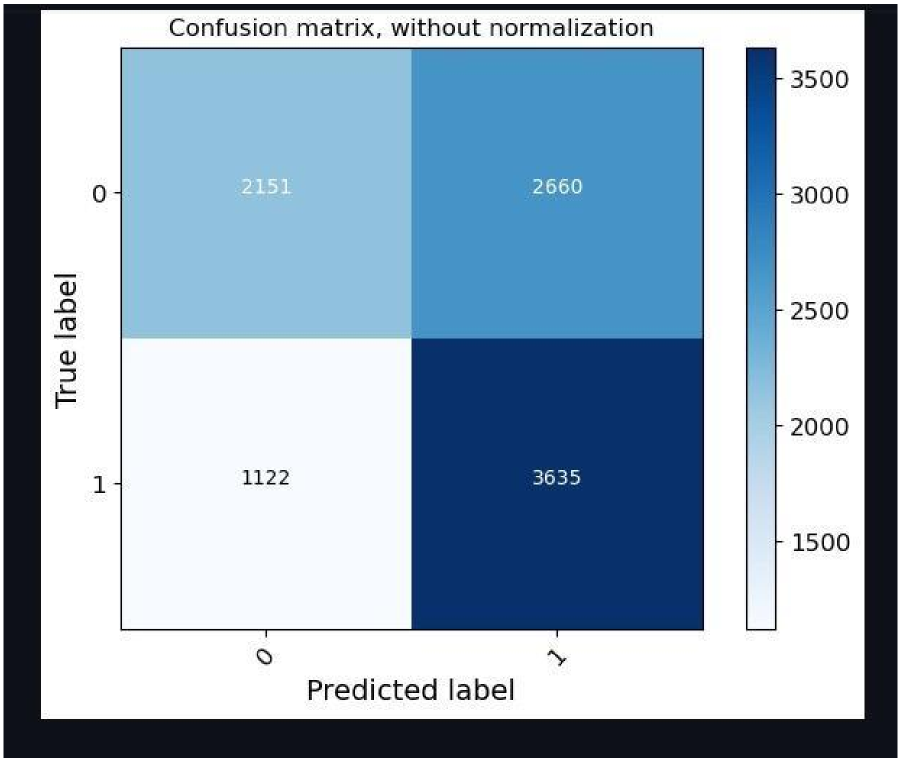
6. Metrics for "Goodness"

- Precision: 0.66 for class 0 and 0.58 for class 1
- Recall: 0.45 for class 0 and 0.76 for class 1
- F1-Score: 0.53 for class 0 and 0.66 for class 1
- Predictive accuracy: 60.4%
- Classification error: 39.6%

```
# Print classification report
print(f"\t\t The classification report  \n\n {report}")
```

✓ 0.0s

```
                  The classification report 

              precision    recall  f1-score   support

           0       0.66      0.45      0.53      4811
           1       0.58      0.76      0.66      4757

    accuracy                           0.60      9568
   macro avg       0.62      0.61      0.59      9568
weighted avg       0.62      0.60      0.59      9568
```

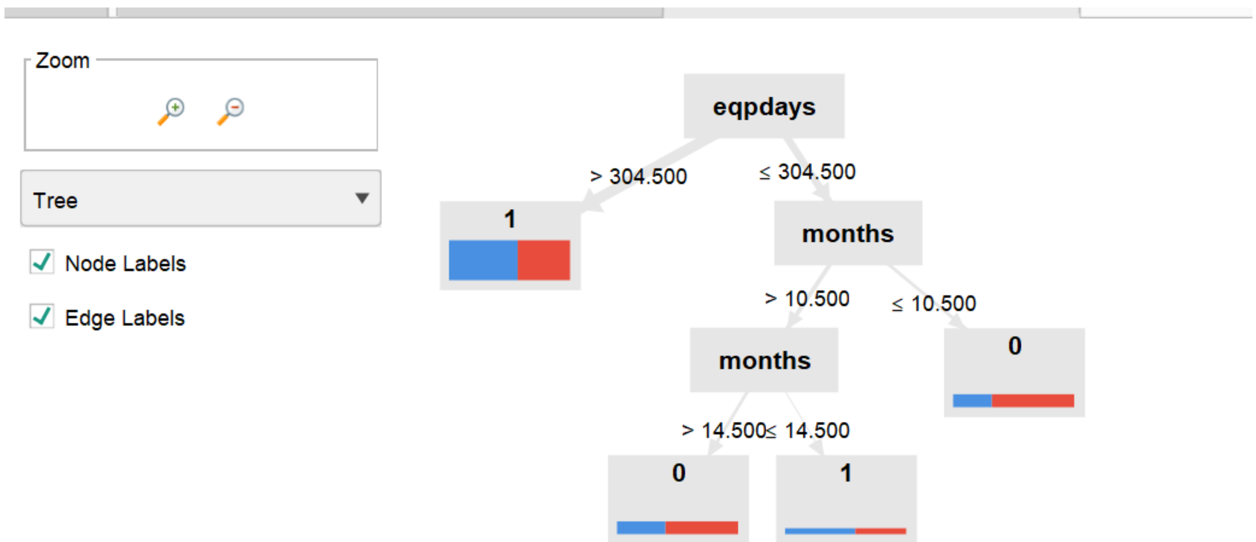


7. Conclusion

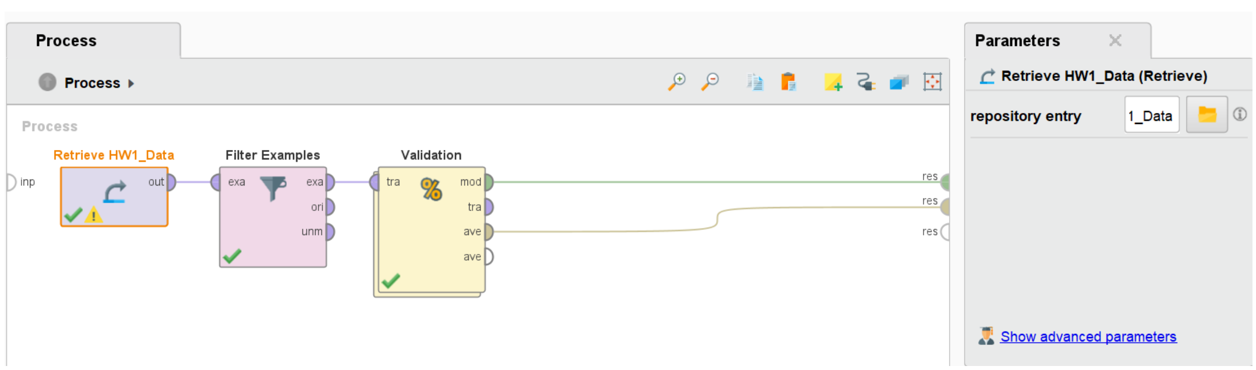
The Decision Tree model with the best parameters achieved an accuracy of 60.4%, which is an improvement over the initial model. This model can be a good starting point for predicting the classification of 'churndep' in our dataset.

20 Points (Rapidminer):
As you discuss the results please make sure you provide screenshots of your corresponding Python code at the same time. At the end, also please provide the Rapidminer screenshots as well (Screenshot on how you split the data, how you built and evaluated the model, the Parameters panel for the Decision tree operator, all the corresponding performance metrics as well as the visualization of the decision tree).

Decision Tree

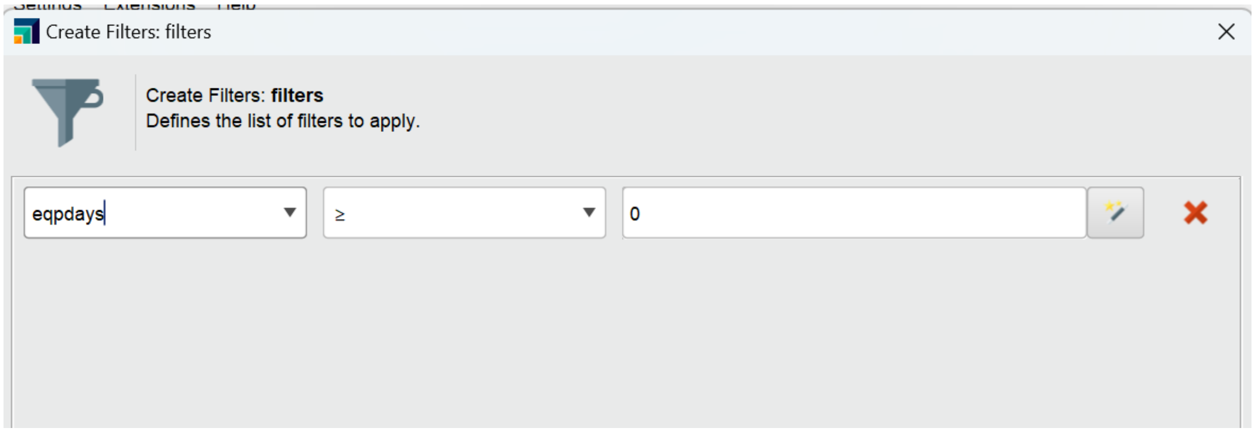


Data



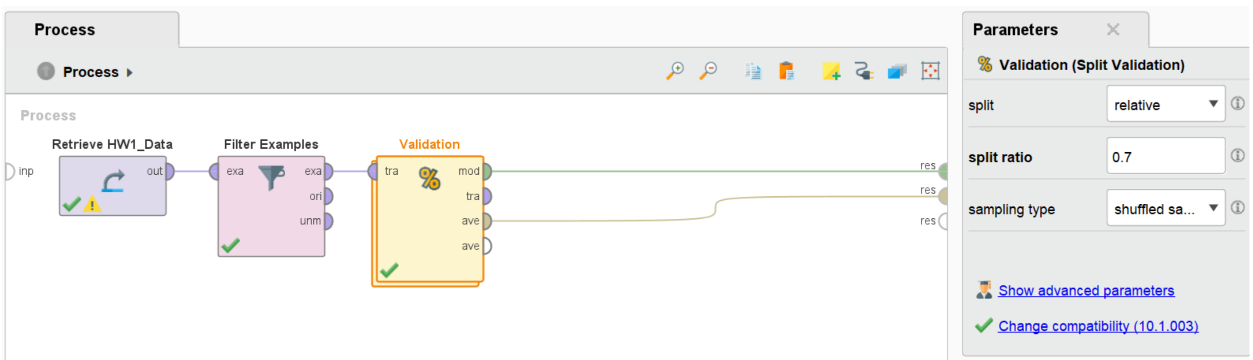
Filter

Epqdays is the number of days the customer has had his/her current equipment. This number can't be negative because customers can't possess the equipment for negative number of days. However, the number could be 0 when customers just have possessed the equipment for less than a day. We set the filter to be greater or equal to 0.



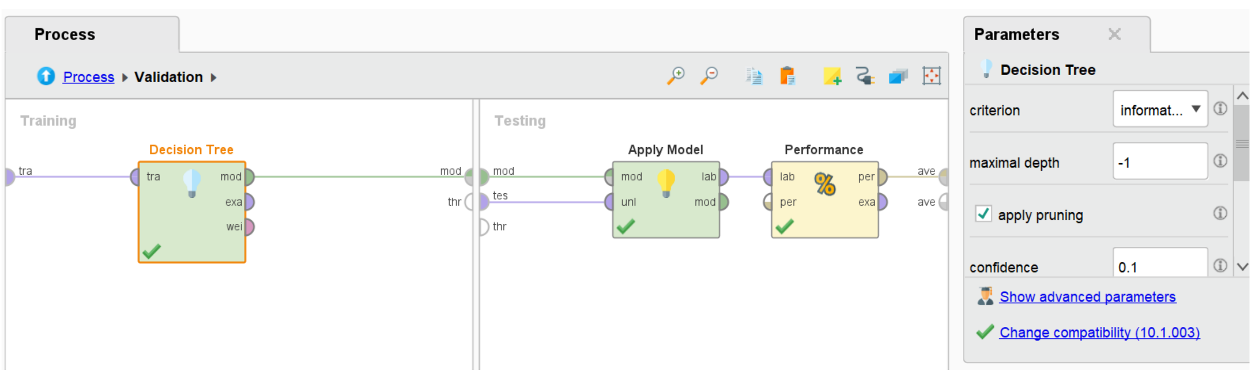
Split Validation

We split the data into training data and test data by the proportion 0.7 and shuffled the data to make it random.



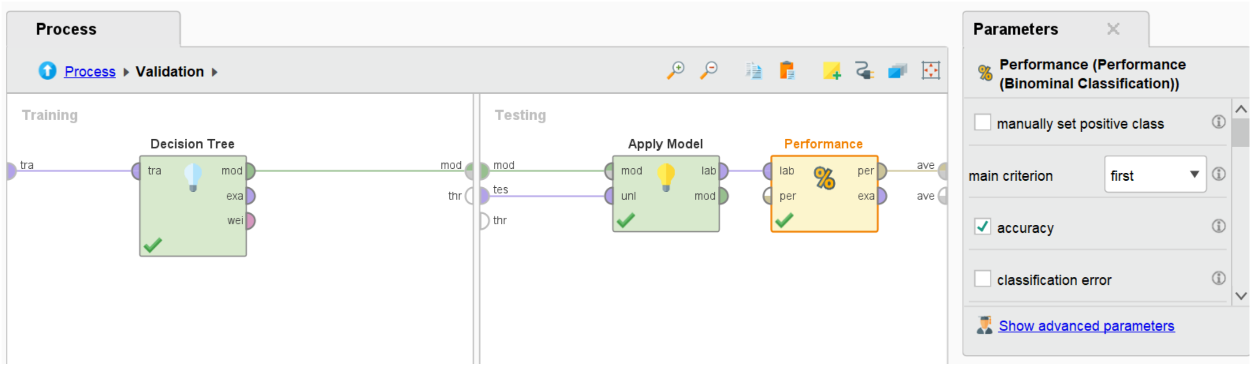
Decision Tree

We need to use entropy as the criterion, so we select information gain.
To put no bound on the maximal depth, we set it to be -1.



Performance

Since the target variable is binary, we choose binomial classification.



Evaluation

accuracy: 59.71%

	true 1	true 0	class precision
pred. 1	3649	2720	57.29%
pred. 0	1129	2055	64.54%
class recall	76.37%	43.04%	

precision: 64.54% (positive class: 0)

	true 1	true 0	class precision
pred. 1	3649	2720	57.29%
pred. 0	1129	2055	64.54%
class recall	76.37%	43.04%	

recall: 43.04% (positive class: 0)

	true 1	true 0	class precision
pred. 1	3649	2720	57.29%
pred. 0	1129	2055	64.54%
class recall	76.37%	43.04%	

f_measure: 51.64% (positive class: 0)

	true 1	true 0	class precision
pred. 1	3649	2720	57.29%
pred. 0	1129	2055	64.54%
class recall	76.37%	43.04%	