IBM Employee Attrition Prediction

Analysis Report

---

1. Introduction

Employees have always been important assets of firms as they create values for the firms and are an integral part of the companies' business operation, exemplified by the setting up of human resource department in almost every big firm. Employee attrition would trigger detrimental consequences toward the firms because of the attrition costs, encompassing training cost of new employees, administration cost, etc. It is, therefore, conceivably conducive to curb high attrition rate by predicting potential attrition among the current employees, and thus befitting measures can accordingly be taken. In light of this problem, we would like to analyze the IBM HR Employee dataset, create prediction models upon it, and compare different data preparation and hyperparameter tuning ways. The goal is to generate a prediction model that can accurately predict employee attrition so as to avoid the above-mentioned problems and is to create a reference for how to build and evaluate prediction models for other employee attrition situations.

2. Data Understanding

Dataset: https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset

First, we are going to show some analysis about the dataset:

- Total number of records: **1470**
- Total number of attributes: **35**
- Supervised learning: attribute "Attrition" given, which is the target variable

By simply looking at the frequencies of classes "No" (1233) and "Yes" (237) in "Attrition", we see that an employee would have 16% probability of attrition meaning that our model performance or accuracy should be higher than 84% (majority voting); it also means the target variable – “Attrition” is imbalanced. Now let’s take a look on the statistics of different attributes in Table 1 in Appendix.

Base on Table 1, we have the following observations on the dataset:

- There are two types of data, numeric and string (which has to be changed into categorical in data preparation part).
- Some attributes are intuitively irrelevant to the prediction, such as “EmployeeCount”, “EmployeeNumber”, “Over18” and “StandardHours”. Later we would perform feature selection to check if these attributes are really meaning less, either by filter methods or embedded methods from various inducers. The wrapper methods would also be carried when the error rate of trained models is calculated.
- Some attributes’ meaning is vague or overlapped with other attributes. For example, it is difficult to understand the meaning and differences of “MonthlyIncome” and “MonthlyRate”.
- Some attributes are skewed (for numeric variables) or are with imbalanced classes.
- Some attributes have only one value or class.

3. Data Preparation

Datatype Changing

There are some string variables in our dataset so the first thing we have to do is to change those variables into categorical. We use <*Edit Metadata*> to select the string variables and convert them into categorical. After the action, we now only have categorical and numeric variables.

Feature Selection

To understand what variables are useful for prediction, another thing we have to deal with is that some columns may be meaningless or helpless to the attrition prediction; we would first perform filter method to have a brief idea on what attributes are very likely informative to be selected as features. We use <*Filter Based Feature Selection*> to help. We adjust the following parameters in it:

- Target Column: “Attrition” – the target variable/label

- Number of desired features: 40 – the number of features returned is larger than 35, meaning that all features will be returned even for variables with 0 scores. Since we cannot predict how many columns are useful, we want to return all columns and then remove least informative ones manually so 40 is just a random number greater than the total number of columns in the dataset.
- Feature scoring method: we will try "Mutual Information" as it supports both numeric and categorical variables in our edited dataset

The result is shown as below (in descending order of score):

Attributes	Score
Attrition	1
JobRole	0.028013
OverTime	0.027667
MonthlyIncome	0.025235
JobLevel	0.023591
TotalWorkingYears	0.023082
YearsAtCompany	0.022778
StockOptionLevel	0.020397
YearsWithCurrManager	0.019996
Age	0.019996
YearsInCurrentRole	0.01765
MaritalStatus	0.014725
BusinessTravel	0.007832
NumCompaniesWorked	0.007715
EnvironmentSatisfaction	0.006955
JobSatisfaction	0.005873
JobInvolvement	0.005721
EducationField	0.005399
DistanceFromHome	0.005117
DailyRate	0.004963
Department	0.003696
MonthlyRate	0.003455
HourlyRate	0.003379

TrainingTimesLastYear	0.003104
PercentSalaryHike	0.00297
YearsSinceLastPromotion	0.002264
WorkLifeBalance	0.002208
EmployeeNumber	0.001856
RelationshipSatisfaction	0.001702
Education	0.0009
Gender	0.000418
PerformanceRating	0.000014
EmployeeCount	0
Over18	0
StandardHours	0

Table 2 – Correlation scores with "Attrition".

We can see "EmployeeCount", "Over18" and "StandardHours" both have score of 0, implying that they do not help in prediction, which matches our expectation above. We then use <*Select Columns in Dataset*> to exclude them; we would also remove "EmployeeNumber" because according to its definition it is just employee's ID. This does not provide any information for prediction. For other features with a very low score, we do not remove them because we save them for double-check using other feature selection methods. Further feature selection would be conducted later in different models as embedded methods or as wrapper methods in evaluation, as the features returned by filter method may be redundant because correlation between features are not considered; the feature selection here is for more understanding on the dataset and first-time selection. Now we have a dataset with 31 columns, all in numeric or categorical.

Noise and Outliers Removal

For numeric variables, we would remove all the noise and outliers in a bid to increase the accuracy of the model. Notice that those numeric variables, which are actually representing some categories, will not be clipped. For example, even though "Education" is numeric, its values indeed represent different education level (refer to Table 1 in Appendix) so we would leave the discretization of the variables to the underlying mechanism of different models. We use <*Clip Values*> to do noise and outliers removal, and set the parameters as following:

- Set of thresholds: "ClipPeaksAndSubpeaks" – set upper and lower clipped values

- Threshold: "Percentile"
- Percentile number of upper threshold: 99 (around 3 standard deviations range)
- Lower threshold: 1 (around 3 standard deviations range)
- Substitute value for peaks/subpeaks: "Missing" – replace outliers with missing value

Originally the dataset does not have any missing values but after we replace the outliers, there should be some missing values, and therefore <*Clean Missing Data*> is used to clean up the missing values in numeric variables except those categorical-like columns, replacing them with the mean.

Feature Normalization/Scaling (of Numeric Variables)

From Table 1, we can see some numeric variables, such as "DistanceFromHome", are skewed. Therefore, the next step is to normalize the skewed features; likewise, those categorical-like variables will be processed later. We use <*Normalize Data*> to help us to achieve the normalization:

- Transformation method: "Zscore"
- Columns to transform: "MonthlyIncome", "TotalWorkingYears", "YearsAtCompany", "YearsWithCurrManager", "Age", "YearsInCurrentRole", "NumCompaniesWorked", "DistanceFromHome", "DailyRate", "MonthlyRate", "HourlyRate", "TrainingTimesLastYear", "PercentSalaryHike", "YearsSinceLastPromotion" – 14 columns in total

Changing Categorical-like Variables

As we mentioned, there are some categorical-like numeric variables, we, again, use <*Edit Metadata*> to make them categorical:

- Categorical: "Make categorical"
- Column: "JobLevel", "StockOptionLevel", "EnvironmentSatisfaction", "JobSatisfaction", "JobInvolvement", "WorkLifeBalance", "Education", "RelationshipSatisfaction", "PerformanceRating" – 9 columns in total are categorical-like

The last step of data preparation and cleansing is to deal with the imbalanced categorical target variable.

Balancing Categorical Target Variable

There are many ways to handle this problem, such as oversampling, undersampling, cluster-based oversampling, informed oversampling and so on. We would select **oversampling** because although it may cause overfitting and increase processing time, it does not trigger information loss like undersampling. Overfitting can be solved by tuning some hyperparameters of the learning models and using other techniques; but it is hard to recover the lost information. Thus, it is better off to choose oversampling.

We use <*Convert to CSV*> to download the processed dataset and then randomly replicate some examples with "Attrition" value equals to "Yes". The original event rate is 16%; after oversampling, the event rate is now 37%.

- Total number of records: **1943**
- Total number of columns: **31**

We do not add a lot of "Yes" examples to completely balance two classes in a bid to avoid overfitting problem. This also means that for our models, we need to achieve at least 63% of accuracy to make the models work.

The procedures to prepare the dataset can be found in Appendix Figure 1.

4. Model Building and Evaluation

Our prediction is a binary classification problem so we would first select the best-performed model.

Splitting the Dataset

We need a total of three datasets: training, validation and testing sets. We first use <*Split Data*> to split the cleaned dataset into two parts:

- <u>Splitting mode</u>: "Split Rows"
- <u>Fraction of rows in the first output dataset</u>: 0.6
- <u>Stratified split</u>: "True" – apportion equally among the two result datasets based on some columns
- <u>Stratification key column</u>: "Attrition" – the target column for apportioning

Stratified split is set to be “True” and the key column is chosen to be “Attrition” because we want the two result datasets to have roughly equal portion of “Yes” and “No” classes of “Attrition”, avoiding the problem that the split causes uneven distribution of the target variable “Attrition”. After the split, we should have two datasets; but we need one more. Therefore, we perform <*Split Data*> on the **right dataset** again with the same settings except that the fraction is changed to 0.5. After the operation, we should now have three sets: 60% for training, 20% for validation and 20% for testing.

Model Selection

Right now we would choose among the following 4 models:

1. **Two-Class Decision Forest with 1 tree** (simple decision tree)
2. **Two-Class Decision Forest with multiple trees** (ensemble learning for decision tree)
3. **Two-Class Boosted Decision Forest (**ensemble learning for decision tree with boosting**)**
4. **Two-Class Logistic Regression**

We drag a <*Two-Class Decision Forest*>, a <*Two-Class Boosted Decision Forest*> and a <*Two-Class Logistic Regression*> to the experiment. Each of them is linked to <*Tune Model Hyperparameters*> with settings as below:

- Sweeping mode: “Entire grid” – each and every combination is tested
- Label column: “Attrition” – the target variable
- Metric for measuring performance for classification: “Accuracy” – the proportion of true results to total cases

<*Two-Class Decision Forest*> settings:

- Resampling method: “Bagging” – each tree is grown on a new sample, creating diversity
- Create trainer mode: “Parameter Range” – tuning hyperparameters with a particular range

All other settings use the default ones. We just want to approximately estimate which model has the best performance. After we have selected a model, we will further fine-tune the hyparameters on that model. Moreover, since the module <*Tune Model Hyperparameters*> will help you calculate the performance with 1, 8 and 32 trees (default), we do not need to use another <*Two-Class Decision Forest*> module to evaluate the accuracy of 1 decision tree.

<*Two-Class Boosted Decision Forest*> setting:

- Create trainer mode: "Parameter Range"

<*Two-Class Logistic Regression*> setting:

- Create trainer mode: "Parameter Range"

After training the models,

Decision Tree	**Minimum number of samples per leaf node**	**Number of random splits per node**	**Maximum depth of the decision trees**	**Number of decision trees**	**Accuracy**
Two-Class Decision Forest with 1 tree	1	128	16	1	0.83705
Two-Class Decision Forest with multiple trees	1	1024	64	32	0.909091

Table 3 – Selection of Decision Tree Models

Boosted Decision Tree	**Number of leaves**	**Minimum leaf instances**	**Learning rate**	**Number of trees**	**Accuracy**
Two-Class Boosted Decision Forest	32	10	0.4	500	0.927959

Table 4 – Selection of Boosted Decision Tree Models

Logistic Regression	**Optimization Tolerance**	**L1Weight**	**L2Weight**	**Memory Size**	**Accuracy**

Two-Class Logistic Regression	0.0001	0	1	50	0.824185

Table 5 – Selection of Logistic Regression Models

From Table 3 and the visualization of the tuning results, we can already exclude **Two-Class Decision Forest with 1 tree** because no matter how the hyparameters are tuned, the best model with 1 tree only ranks 25.

Having the 3 models with different types of inducer, we can now validate the models using <*Cross Validate Model*> with "Attrition" as the Label Column, connecting to <*Evaluate Model*>. Connect the validation dataset (from the data split) to the validation modules and run them.

	Accuracy
Two-Class Decision Forest with 32 trees	0.846
Two-Class Boosted Decision Forest	0.861
Two-Class Logistic Regression	0.820

Table 6 – Validation Results

So now we would select **Two-Class Boosted Decision Forest** as our model because it generates the highest accuracy. It is conceivable because boosting provides "feedback" to the next classifier so it generally performs better.

Finally, we test the model by connecting the <*Tune Model Hyperparameters*> of trained **Two-Class Boosted Decision Forest** and the testing dataset from the data split to <*Score Model*> and then to <*Evaluate Model*>.

Actual\Prediction	+	**-**
+	116	26
–	2	244
	Accuracy:	0.928

Table 7 – Confusion Matrix and Accuracy of Testing Result

Fine-Tuning Hyperparameters

Above we use the default settings of <*Tune Model Hyperparameters*> Parameter Range; the set of hyperparameters creating "the best model" is shown in Table 4. We now base on that hyperparameter set to further fine-tune hyperparameters to see if the performance can be improved. The same testing set should be used to evaluate the model so as to compare its performance with the selected model shown in Table 4. We add another <*Two-Class Boosted Decision Forest*> module trained by <*Train Model*> with "Attrition" as the target variable and supplied with the training set. Similarly, we evaluate the model with <*Score Model*> and <*Evaluate Model*>. This time, we set the Create trainer mode to be "Single Parameter" and manually tune the hyperparameters one by one, with reference to the hyperparameter set in Table 4. Each time we only tune one hyperparameter.

Original (Default) Settings when "Single Parameter" is chosen:

Maximum number of leaves per tree	Minimum number of samples per leaf node	Learning rate	Number of trees constructed	Accuracy
20	10	0.2	100	0.912

Table 8 – Original Hyperparameter Set

Value	Accuracy
32	0.928
40	0.925
20	0.912
30	0.918
35	0.920

Table 9 – Tuning "Maximum number of leaves per tree"

We, therefore, change the "Maximum number of leaves per tree" to **32** and then tune "Minimum number of samples per leaf node".

Value	Accuracy
15	0.925
5	0.923
12	0.928
11	0.920
9	0.920

Table 10 – Tuning "Minimum number of samples per leaf node"

We thus change the "Minimum number of samples per leaf node" to **12** and then tune "Learning rate".

Value	Accuracy
0.4	0.920
0.5	0.925
0.6	0.925
0.25	0.912
0.15	0.918
0.05	0.899

Table 11 – Tuning "Learning rate"

We thus change the "Learning rate" to **0.2** and then tune "Number of trees constructed".

Value	Accuracy
500	0.920
400	0.923
200	0.920
450	0.918
350	0.925
50	0.920

Table 12 – Tuning "Number of trees constructed"

We thus change the "Number of trees constructed" to **100**.

Finally, we use two <*Add Rows*> to combine the training, validation and testing datasets and then train the model with the whole dataset. For the whole procedures for Model Training and Evaluation, please refer to Figure 2 in Appendix and the comment of the modules in the picture.

5. Conclusion

After training and evaluating different models with different hyperparameters, we conclude that **Two-Class Boosted Decision Forest** can build the best model among other classifiers or models. Tuning the hyperparameters one by one, we can see, in our experiment, there are more than one set of hyperparameters can give an accuracy of 92.8%; this number is very likely to be the best performance of the model, which is higher than 86% (refer to Data Understanding part). Therefore, our model works.

On the other hand, in the Feature Selection part, we eliminate 4 columns; but there are some columns with a very low "Mutual information" score that we did not remove during Data Preparation. We would like to see what the meaningful and pertinent features are. If we visualize the <*Train Model*> or <*Tune Model Hyperparameters*> of **Two-Class Boosted Decision Forest** and check the graphs of the trees, we would have an intuition that some features are more important in determining the attrition of employees (in descending order of the importance): "OverTime", "JobLevel", "Age", "YearsAtCompany", etc. Some less important feature: "Educaton", "EducationField", etc. For future work, ones may select some sets of columns to build the model to validate what features can give a more fruitful model performance (wrapper).

This generally matches our correlation analysis done at the beginning; but note that some features are not embedded to build the trees, and definitely the model would give a more accurate view on the features' importance. More importantly, firms are now able to know the factors that cause employee attrition and accordingly take corresponding measures to avoid problems invoked by attrition. We also hope our project would help firms to create a reference of how to build and evaluate prediction models for other employee attrition situations.

6. Reference

https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/filter-based-feature-selection

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/clip-values

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/normalize-data

https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/tune-model-hyperparameters

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-decision-forest

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-logistic-regression

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/data-transformation-manipulation

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/cross-validate-model

7. Appendix

Table 1 – Statistics about the attributes.

Attributes	Description	Data Type	Missing Values / Unique Values/SD	Mean / Min	Median / Max	Skewed/ Class Imbalance
Age	Age of the employee	Numeric	0	36.93	36	-
			43 / 9.14	18	60	
Attrition	Employee attrition or not	String	0	-		More class "No"
			2 / -			
BusinessTravel	How often the employee travels	String	0	-		More class "Travel_Rarely"
			3 / -			
DailyRate	Unclear	Numeric	0	802.49	802	-
			886 / 403.51	102	1499	
Department	Department of the employee	String	0	-		More class "R&D"
			3 / -			
DistanceFromHome	Intuitive	Numeric	0	9.19	7	Skewed to close distance
			29 / 8.11	1	29	
Education	1 'Below College' 2 'College' 3 'Bachelor' 4 'Master' 5 'Doctor'	Numeric	0	2.91	3	Skewed to '3'
			5 / 1.02	1	5	
EducationField	Intuitive	String	0	-		More class "Life Sciences"
			6 / -			
***EmployeeCount**	Unclear	Numeric	0	1	1	Only one value '1'
			1 / 0	1	1	
***EmployeeNumber**	Employee ID	Numeric	0	1024.86	1020.5	Unique for each employee
			1470 / 602.02	1	2068	
EnvironmentSatisfaction	1 'Low' 2 'Medium'	Numeric	0	2.72	3	Skewed to '3' and '4'

	3 'High' 4 'Very High'		4 / 1.09	1	4	
Gender	Intuitive	String	0	-		More class “Male”
			2 / -			
HourlyRate	Unclear	Numeric	0	65.89	66	2 modals
			71 / 20.33	30	100	
JobInvolvement	1 'Low' 2 'Medium' 3 'High' 4 'Very High'	Numeric	0	2.73	3	Skewed to ‘3’
			4 / 0.71	1	4	
JobLevel	Unclear (categorical-like)	Numeric	0	2.06	2	Skewed to ‘1’ and ‘2’
			5 / 1.11	1	5	
JobRole	The type of employees’ job	String	0	-		More class “Sales Executive”
			9 / -			
JobSatisfaction	1 'Low' 2 'Medium' 3 'High' 4 'Very High'	Numeric	0	2.73	3	Skewed to ‘3’ and ‘4’
			4 / 1.10	1	4	
MaritalStatus	Intuitive	String	0	-		More class “Married”
			3 / -			
MonthlyIncome	Unclear compared to MonthlyRate	Numeric	0	6502.93	4919	Skewed to low income
			1349 / 4707.96	1009	19999	
MonthlyRate	Unclear	Numeric	0	14313.1	14235.5	-
			1427 / 7117.9	2094	26999	
NumCompaniesWorked	Intuitive	Numeric	0	2.69	2	Skewed to ‘2’
			10 / 2.50	0	9	
***Over18**	Over 18 years old	String	0	-		One unique class
			1 / -			
OverTime	Over time working	String	0	-		More class “No”
			2 / -			
PercentSalaryHike	Wage percent increase	Numeric	0	15.21	14	Skewed to ‘11’
			15 / 3.66	11	25	
PerformanceRating	1 'Low' 2 'Good' 3 'Excellent'	Numeric	0	3.15	3	Skewed to ‘3’ and ‘4’
			2 / 0.36	3	4	

	4 'Outstanding'					
RelationshipSatisfaction	1 'Low' 2 'Medium' 3 'High' 4 'Very High'	Numeric	0	2.71	3	-
			4 / 1.08	1	4	
***StandardHours**	Standard Working Hrs	Numeric	0	80	80	One Unique Value '80'
			1 / 0	80	80	
StockOptionLevel	Unclear (categorical-like)	Numeric	0	0.79	1	Skewed to '1' and '2'
			4 / 0.85	0	3	
TotalWorkingYears	Years the employee has been working	Numeric	0	11.28	10	Skewed to lower years
			40 / 7.78	0	40	
TrainingTimesLastYear	Intuitive	Numeric	0	2.80	3	Skewed to medium hours
			7 / 1.29	0	6	
WorkLifeBalance	1 'Bad' 2 'Good' 3 'Better' 4 'Best'	Numeric	0	2.76	3	Skewed to '3'
			4 / 0.71	1	4	
YearsAtCompany	Years stayed in IBM	Numeric	0	7.01	5	Skewed to lower years
			37 / 6.13	0	40	
YearsInCurrentRole	Intuitive	Numeric	0	4.23	3	Skewed to lower years
			19 / 3.62	0	18	
YearsSinceLastPromotion	Intuitive	Numeric	0	2.19	1	Skewed to lower years
			16 / 3.22	0	15	
YearsWithCurrManager	Intuitive	Numeric	0	4.12	3	Skewed to lower years
			18 / 3.57	0	17	

* rows in red are removed during the feature selection step

Figure 1 – Procedures for Data Preparation

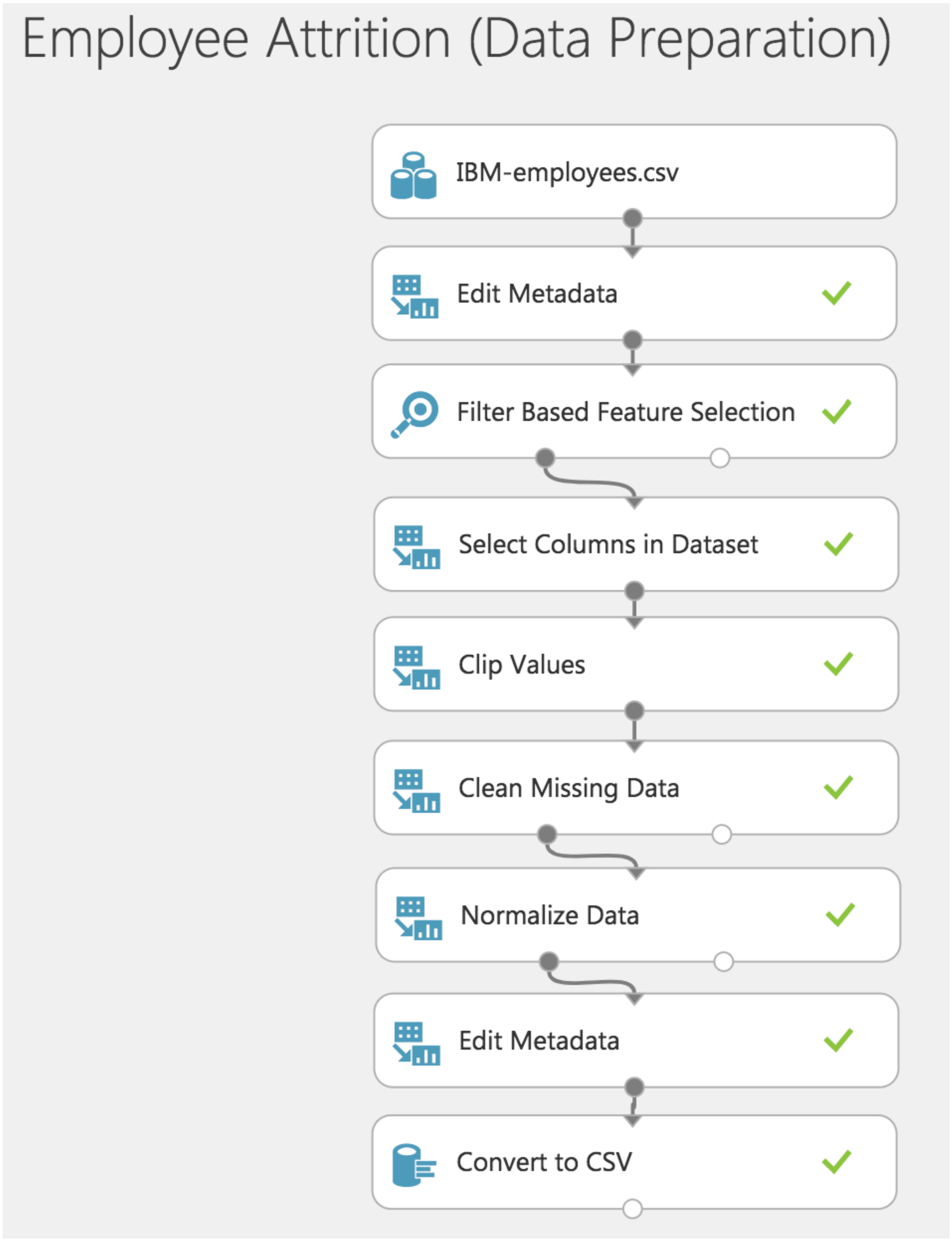
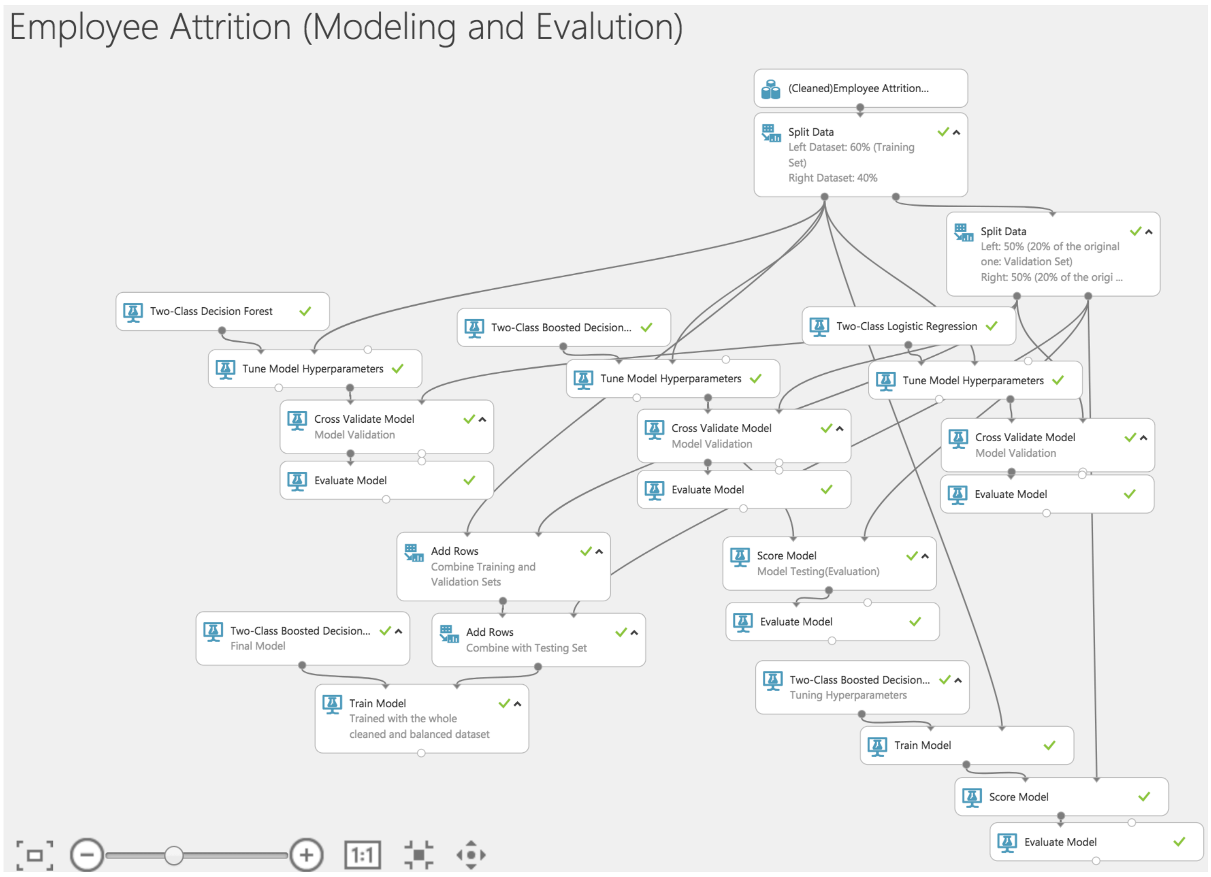


Figure 2 – Whole procedures for Modeling and Evaluation


Employee Attrition (Modeling and Evalution)
(Cleaned)Employee Attrition...
Split Data
Left Dataset: 60% (Training Set)
Right Dataset: 40%
Split Data
Left: 50% (20% of the original one: Validation Set)
Right: 50% (20% of the origi ...
Two-Class Decision Forest
Two-Class Boosted Decision...
Two-Class Logistic Regression
Tune Model Hyperparameters
Tune Model Hyperparameters
Tune Model Hyperparameters
Cross Validate Model
Model Validation
Cross Validate Model
Model Validation
Cross Validate Model
Model Validation
Evaluate Model
Evaluate Model
Evaluate Model
Add Rows
Combine Training and Validation Sets
Score Model
Model Testing(Evaluation)
Evaluate Model
Two-Class Boosted Decision...
Final Model
Add Rows
Combine with Testing Set
Two-Class Boosted Decision...
Tuning Hyperparameters
Train Model
Trained with the whole cleaned and balanced dataset
Train Model
Score Model
Evaluate Model
1:1