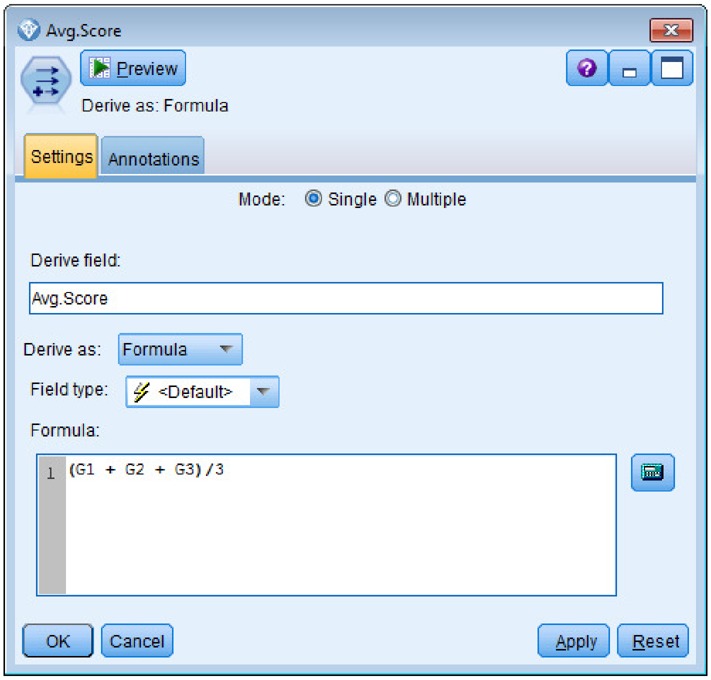
Step by Step Example – Using Gradient-Boosted Trees for Regression

For this example, we will attempt predicting the average grades for students based on a number of variables. The dataset used for this example can be found at the UCI MachineLearning Repository [1]. This dataset can be found at http://archive.ics.uci.edu/ml/datasets/Student+Performance. The data was originally used by Paulo Cortez [2].

This is a toy example to demonstrate how this extension can be used locally to use Gradient Boosted Trees for smaller datasets, as well as in a Spark environment using Analytic Server.

1. Download data and open in Modeler using a Var File node. This file is ';' (semi-colon) separated, so you will need to mark other for type of delimiter and type in a semi-colon.
2. This dataset provides 3 different grade response variables. This is more than we are interested in for now, so rather than just predicting one, let's average the scores as the target we want to predict.
   a. To do this, add a derive node following the var.file node.
   b. Add the following formula to average the scores.



Derive as: Formula

Mode: ● Single ○ Multiple

Derive field: Avg.Score

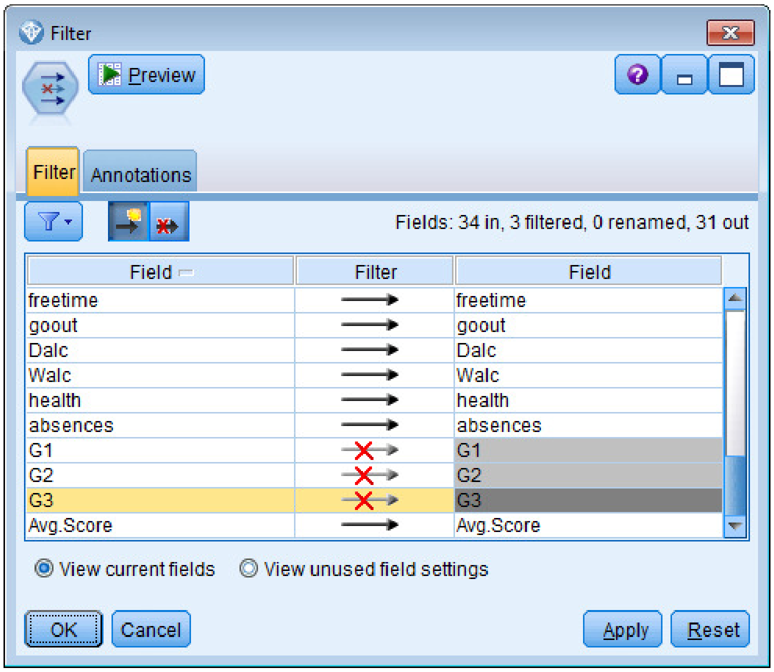
Derive as: Formula

Field type: <Default>

Formula:

```
(G1 + G2 + G3)/3
```

3. Now , let’s filter out the 3 columns for the original grades provided:

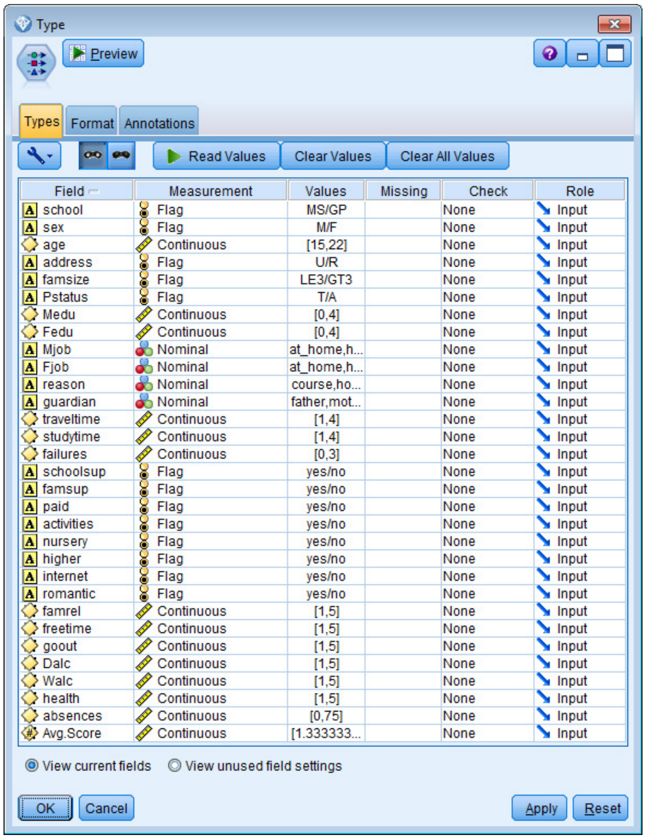


Fields: 34 in, 3 filtered, 0 renamed, 31 out

Field	Filter	Field
freetime	→	freetime
goout	→	goout
Dalc	→	Dalc
Walc	→	Walc
health	→	health
absences	→	absences
G1	✗→	G1
G2	✗→	G2
G3	✗→	G3
Avg.Score	→	Avg.Score

● View current fields ○ View unused field settings

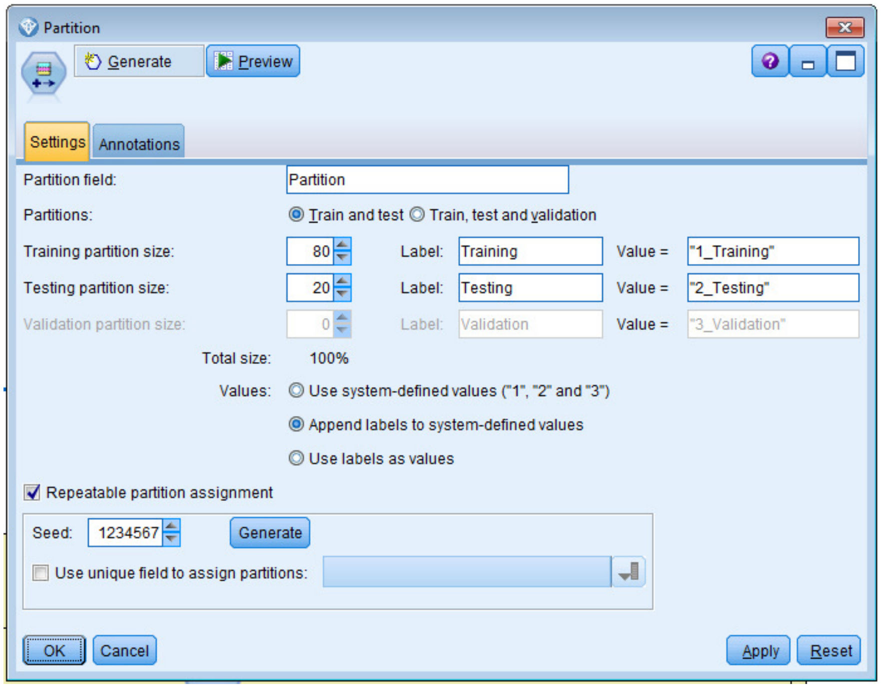
4. Next we can add a Type node, to read values to make sure all the meta-data is correct.



Field	Measurement	Values	Missing	Check	Role
school	Flag	MS/GP		None	Input
sex	Flag	M/F		None	Input
age	Continuous	[15,22]		None	Input
address	Flag	U/R		None	Input
famsize	Flag	LE3/GT3		None	Input
Pstatus	Flag	T/A		None	Input
Medu	Continuous	[0,4]		None	Input
Fedu	Continuous	[0,4]		None	Input
Mjob	Nominal	at_home,h...		None	Input
Fjob	Nominal	at_home,h...		None	Input
reason	Nominal	course,ho...		None	Input
guardian	Nominal	father,mot...		None	Input
traveltime	Continuous	[1,4]		None	Input
studytime	Continuous	[1,4]		None	Input
failures	Continuous	[0,3]		None	Input
schoolsup	Flag	yes/no		None	Input
famsup	Flag	yes/no		None	Input
paid	Flag	yes/no		None	Input
activities	Flag	yes/no		None	Input
nursery	Flag	yes/no		None	Input
higher	Flag	yes/no		None	Input
internet	Flag	yes/no		None	Input
romantic	Flag	yes/no		None	Input
famrel	Continuous	[1,5]		None	Input
freetime	Continuous	[1,5]		None	Input
goout	Continuous	[1,5]		None	Input
Dalc	Continuous	[1,5]		None	Input
Walc	Continuous	[1,5]		None	Input
health	Continuous	[1,5]		None	Input
absences	Continuous	[0,75]		None	Input
Avg.Score	Continuous	[1.333333...		None	Input

● View current fields ○ View unused field settings

5. With our data prepared, let’s partition and split into training and testing
    a. Add a Partition node from the Field Ops Palette
    b. I chose to do 80/20 train/test split for this data.



Partition field: Partition

Partitions: ● Train and test ○ Train, test and validation

Training partition size:	80	Label:	Training	Value =	"1_Training"
Testing partition size:	20	Label:	Testing	Value =	"2_Testing"
Validation partition size:	0	Label:	Validation	Value =	"3_Validation"

Total size: 100%

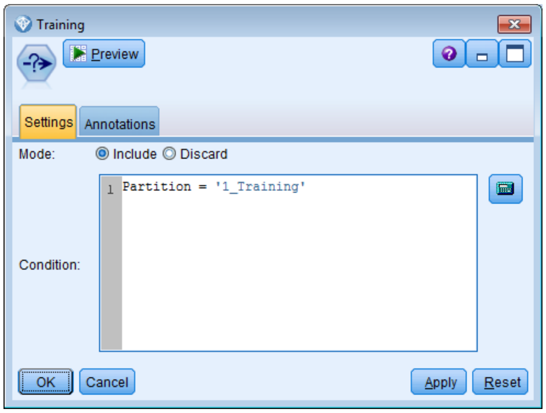
Values: ○ Use system-defined values ("1", "2" and "3")
● Append labels to system-defined values
○ Use labels as values

☑ Repeatable partition assignment

Seed: 1234567

□ Use unique field to assign partitions:

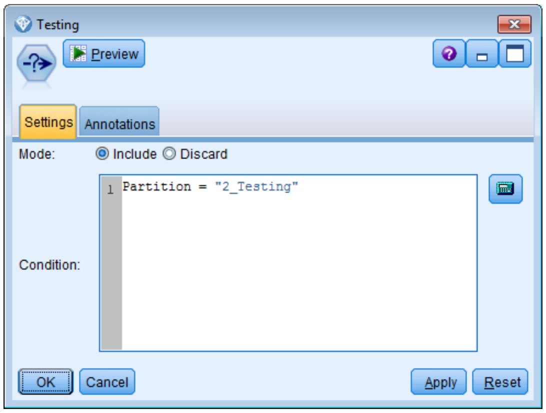
    c. This node will create a new column that gives a field to select training and testing from (the values in this column are “1_Training” and “2_Testing”)
    d. To split the data, add two Select nodes from the Record Ops palette where the formula should match the Nodes below



Mode: ● Include ○ Discard

Condition:

```
Partition = '1_Training'
```

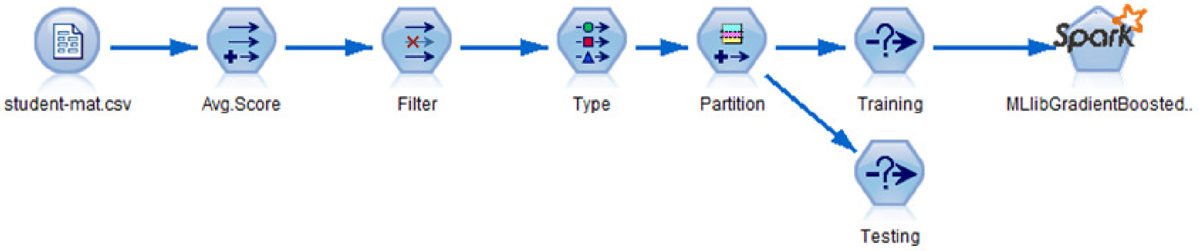


Mode: ● Include ○ Discard

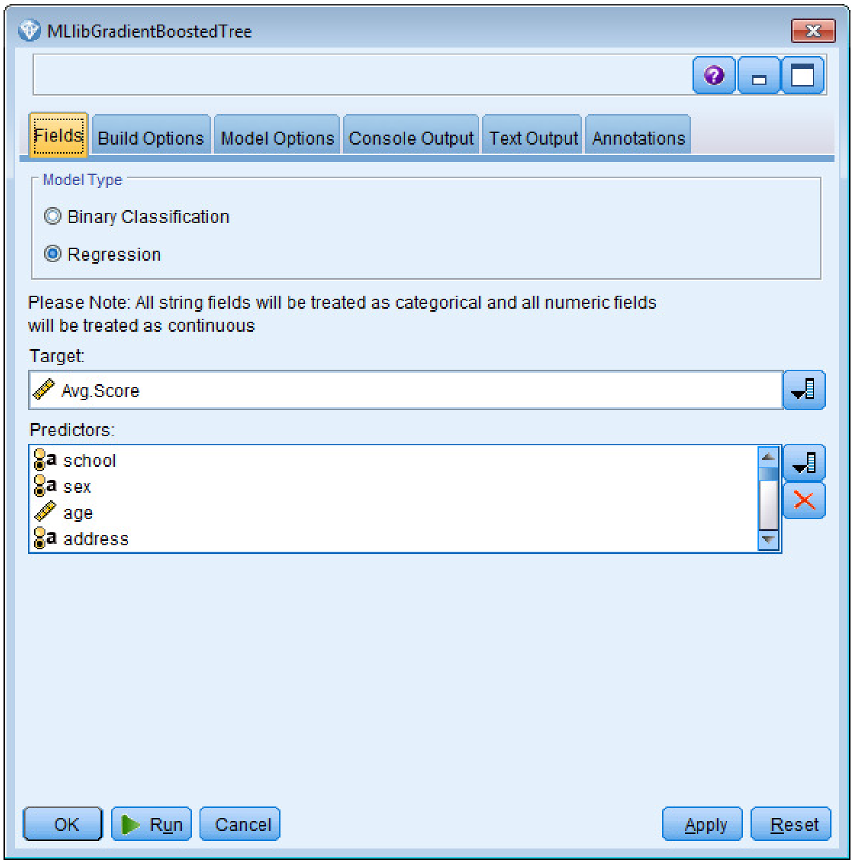
Condition:

```
Partition = "2_Testing"
```

6. Now, let's add our Gradient-Boosted Tree node so our stream will look like this



7. Open the GBT dialog
    a. Make sure Model Type is Regression since we are predicting a numeric variable
    b. Add the Avg. Score as the Target variable
    c. For predictors you can include all variables except Avg.Score and the Partition field



Model Type

○ Binary Classification

● Regression

Please Note: All string fields will be treated as categorical and all numeric fields will be treated as continuous

Target:

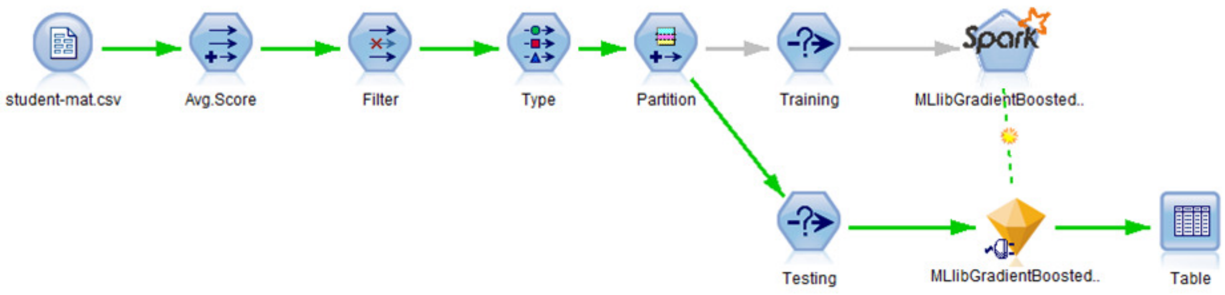
Avg.Score

Predictors:

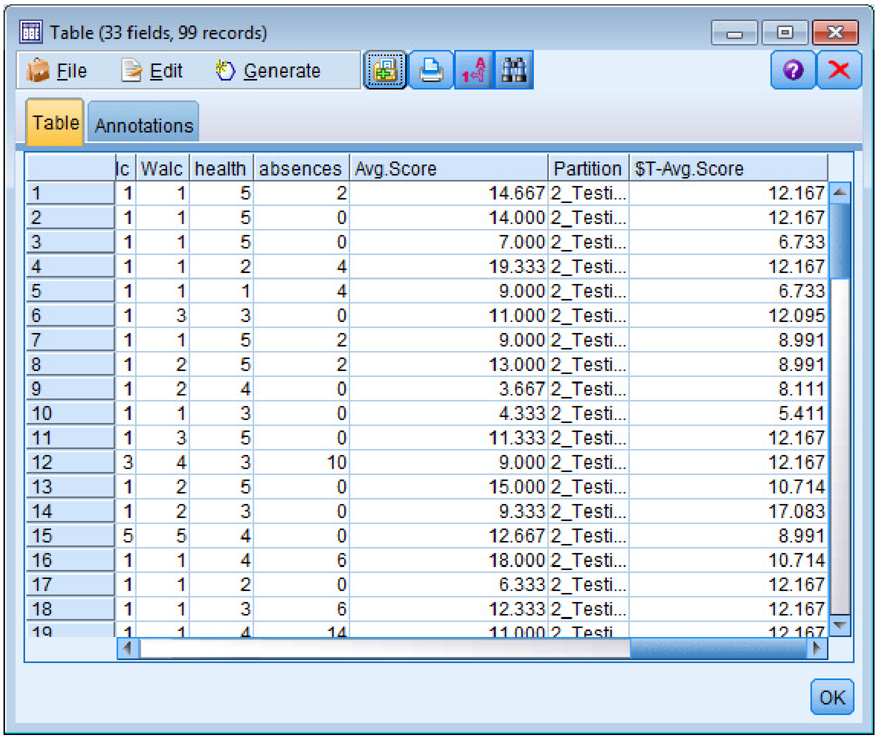
school
sex
age
address

    d. If you would like to fine tune the model, go to Build Options and tweak the parameters.

8. Run this stream to create the model.
9. Now connect the "Testing" Select node to the model nugget create and add a table as the output
10. Right click the table node and run to get the following stream:

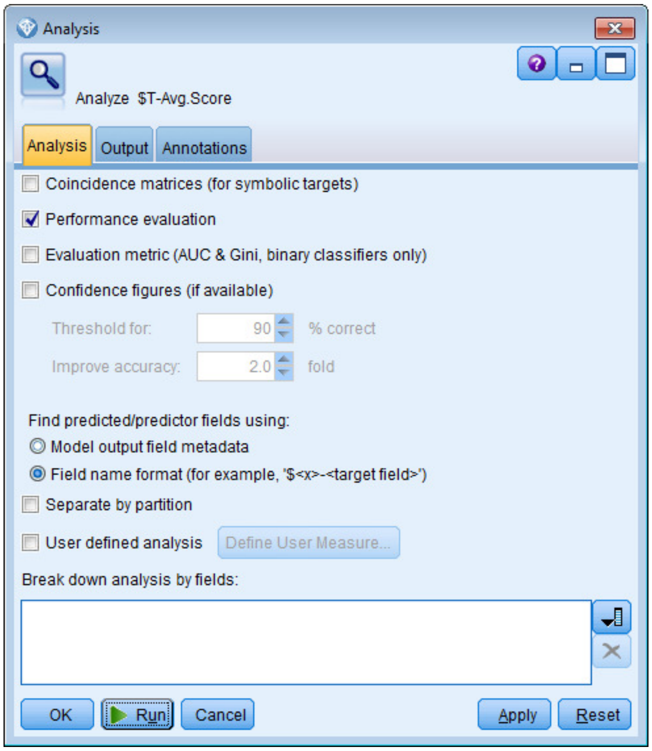


11. The table produced will include the predicted score for each student:



	lc	Walc	health	absences	Avg.Score	Partition	$T-Avg.Score
1	1	1	5	2	14.667	2_Testi...	12.167
2	1	1	5	0	14.000	2_Testi...	12.167
3	1	1	5	0	7.000	2_Testi...	6.733
4	1	1	2	4	19.333	2_Testi...	12.167
5	1	1	1	4	9.000	2_Testi...	6.733
6	1	3	3	0	11.000	2_Testi...	12.095
7	1	1	5	2	9.000	2_Testi...	8.991
8	1	2	5	2	13.000	2_Testi...	8.991
9	1	2	4	0	3.667	2_Testi...	8.111
10	1	1	3	0	4.333	2_Testi...	5.411
11	1	3	5	0	11.333	2_Testi...	12.167
12	3	4	3	10	9.000	2_Testi...	12.167
13	1	2	5	0	15.000	2_Testi...	10.714
14	1	2	3	0	9.333	2_Testi...	17.083
15	5	5	4	0	12.667	2_Testi...	8.991
16	1	1	4	6	18.000	2_Testi...	10.714
17	1	1	2	0	6.333	2_Testi...	12.167
18	1	1	3	6	12.333	2_Testi...	12.167
19	1	1	4	14	11.000	2_Testi...	12.167

12. That is good, but we want to see how the model performed. Add an Analysis node from the Output palette and use the setting below, then run the stream.



Analysis

Analyze $T-Avg.Score

□ Coincidence matrices (for symbolic targets)

☑ Performance evaluation

□ Evaluation metric (AUC & Gini, binary classifiers only)

□ Confidence figures (if available)

Threshold for: 90 % correct

Improve accuracy: 2.0 fold

Find predicted/predictor fields using:

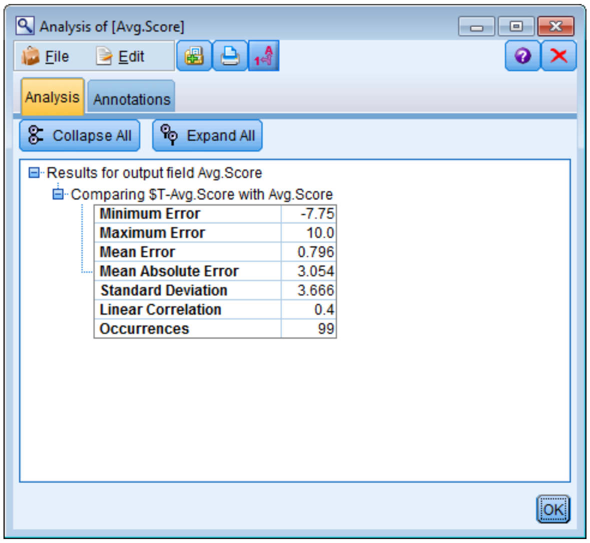
○ Model output field metadata

● Field name format (for example, '$<x>-<target field>')

□ Separate by partition

□ User defined analysis Define User Measure...

Break down analysis by fields:



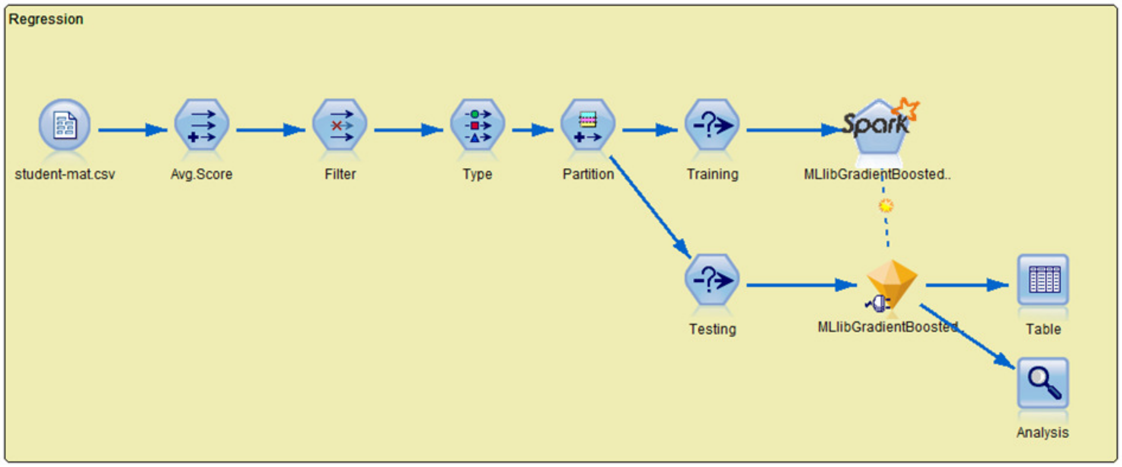
Analysis of [Avg.Score]

Results for output field Avg.Score

Comparing $T-Avg.Score with Avg.Score

Minimum Error	-7.75
Maximum Error	10.0
Mean Error	0.796
Mean Absolute Error	3.054
Standard Deviation	3.666
Linear Correlation	0.4
Occurrences	99

13. Now you have the Mean Error, MAE, and other metrics that will help evaluate the performance of the model.
14. Finally – our full stream should look like this:



References:

[1] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.