Multinomial Naïve Bayes with MLlib

Use the MLlib implementation of Multinomial Naïve Bayes with MLlib to classify your data

Product: IBM® SPSS® Modeler

Extension type: Analysis

Table of Contents

Description:

Naive Bayes is a probabilistic classification algorithm with an assumption of conditional independence between every pair of features. The multinomial Naïve Bayes algorithm implemented in MLlib and used for this extension performs very well in the task of document classification or spam filtering. For this extension, predictors should represent a frequency of a given feature. An example of this would be using a term document matrix where a cell in the matrix would represent the frequency of a given word in a given document.

Requirements:

- SPSS Modeler v18.0 or later
- Python 2.7 Anaconda distribution

Installation:

Initial one-time set-up for PySpark Extensions

If using v18.0 of SPSS Modeler, navigate to the options.cfg file (Windows default path: C:\Program Files\IBM\SPSS\Modeler\18.0\config). Open this file in a text editor and paste the following text at the bottom of the document:

eas_pyspark_python_path, "C:/Users/IBM_ADMIN/Anaconda/python.exe"

The underlined path should be replaced with the path to your python.exe from your Anaconda installation.

Extension Hub Installation

1. Go to the Extension menu Modeler and click "Extension Hub"
2. In the search bar, type the name of this extension and press enter
3. Check the box next to "Get extension" and click OK at the bottom of the screen
4. The extension will install and a pop-up will show what palette it was installed

Manual Installation

1. Save the .mpe file to your computer
2. In Modeler, click the Extensions menu, then click Install Local Extension Bundle
3. Navigate to where the .mpe was saved and click open
4. The extension will install and a pop-up will show what palette it was installed

Python Libraries used:

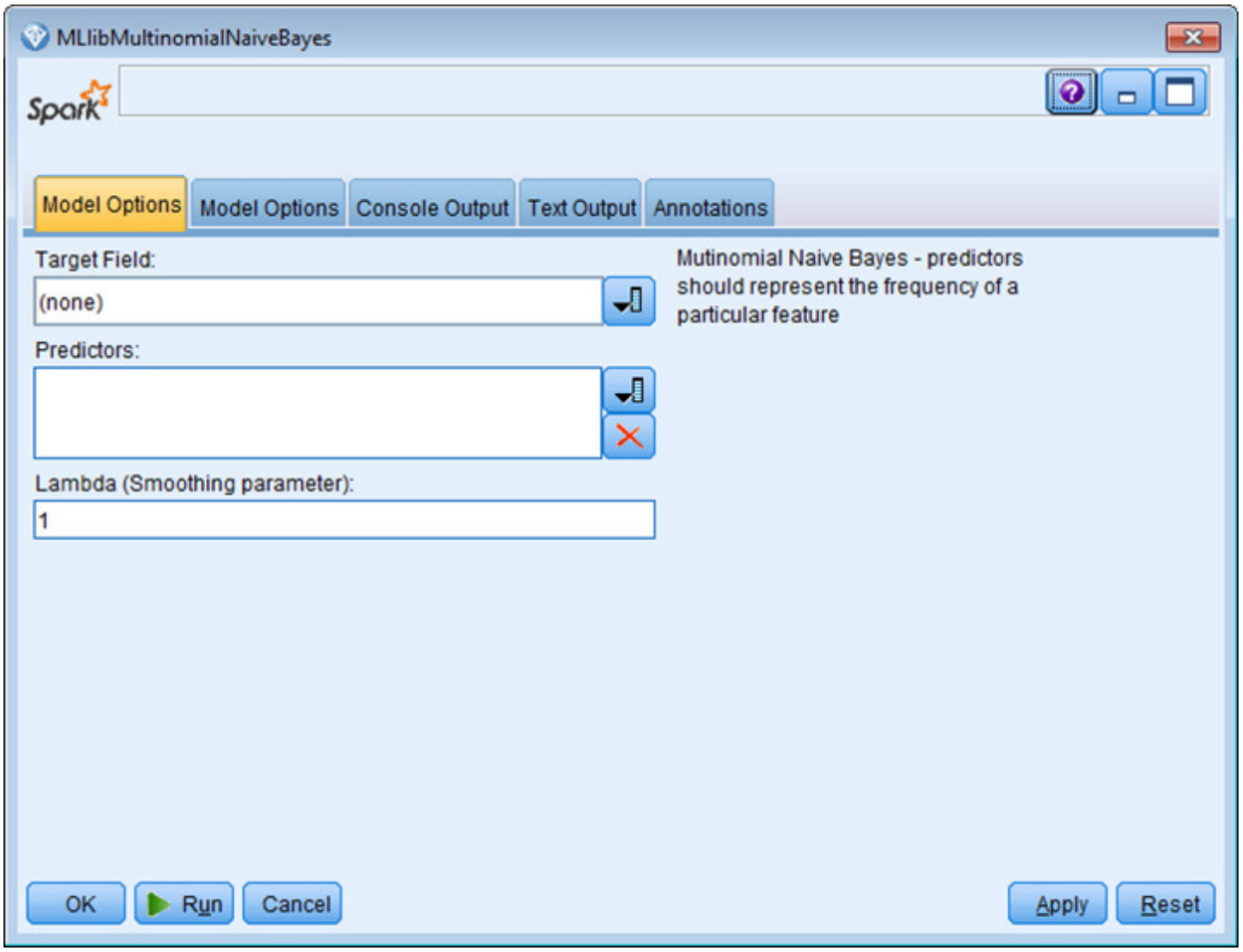
- Spark MLlib – Naive Bayes

User Interface

This extension has a model options tab which allows you to control the features used by the model as well as additional parameters that can be tuned to optimize performance.

Model Options Tab

- Target Field – This is the label or dependent variable you are trying to predict. This should be a string type since the extension is performing classification.
- Predictors – the fields used to predict the target. These fields should be numeric to represent frequency
- Lambda (Smoothing parameter) – default is 1 to perform Laplace smoothing

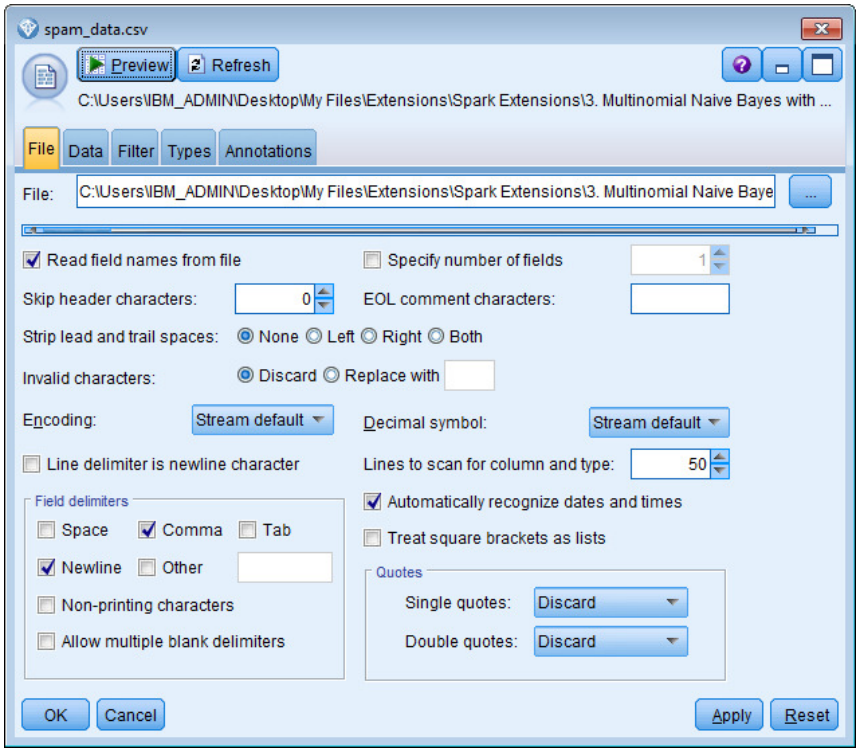


Example:

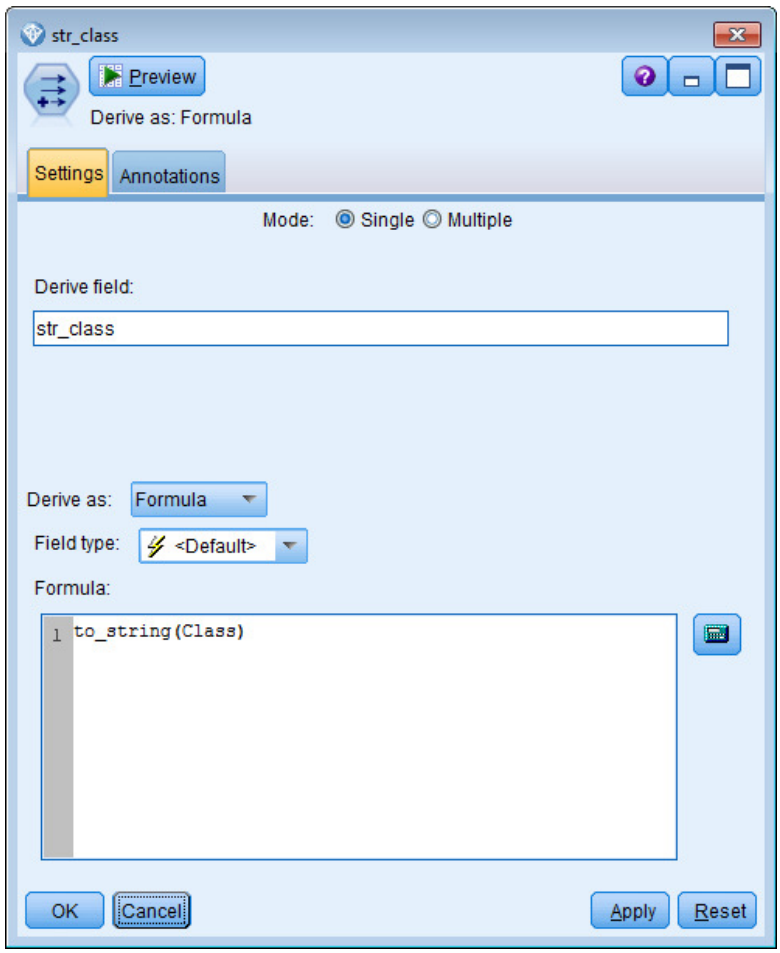
For this example, we will attempt classifying documents (emails) as being spam or not based on a pre-processed term document matrix. The dataset used for this example can be found at the UCI Machine Learning Repository [2]. Please visit the Machine Learning Repository for more information on the dataset. The data was created by the Hewlett-Packard Labs and donated by George Forman [3].

This is an example to demonstrate how this extension can be used locally to use Multinomial Naïve Bayes for document classification. This extension can also be ran with Analytic Server using a cluster of machines to improve performance.

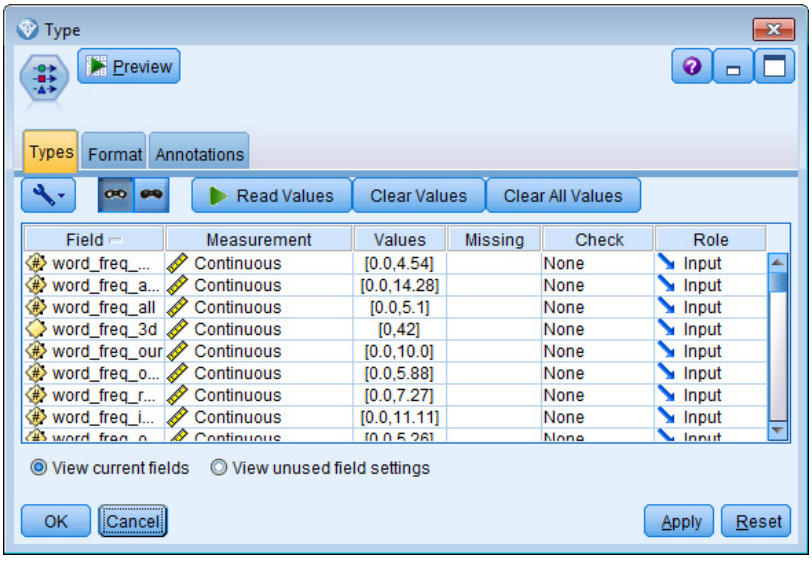
1. Download the data either from UCI or from the GitHub repository in the example directory and open in Modeler using a Var File node.



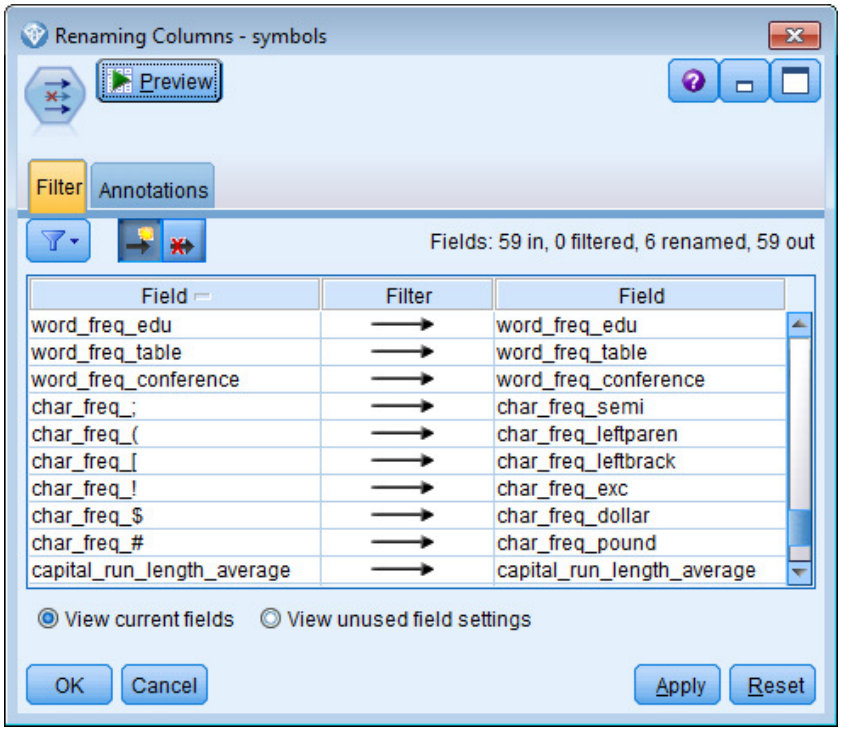
2. This dataset has 58 fields and 4,601 records. In order to classify our class label we need to convert it from 0/1 to '0' & '1'.
    a. To do this, add a derive node following the var.file node.
    b. Add the following formula to convert the class to string.



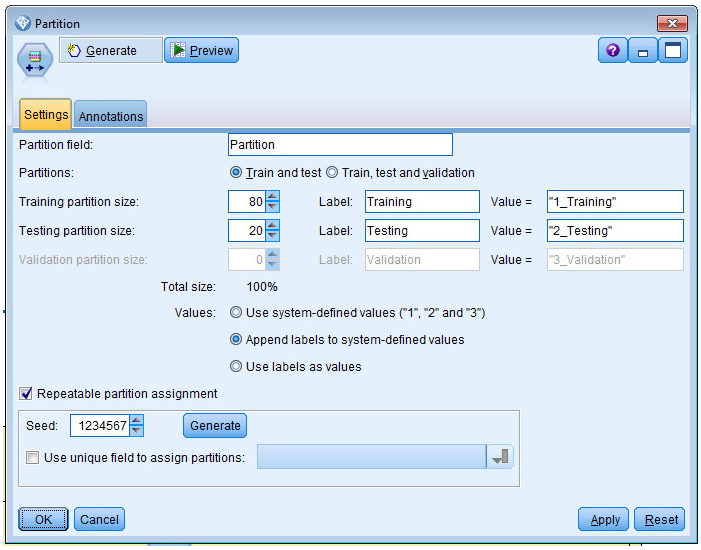
3. Next we can add a Type node, to read values to make sure all the meta-data is correct.



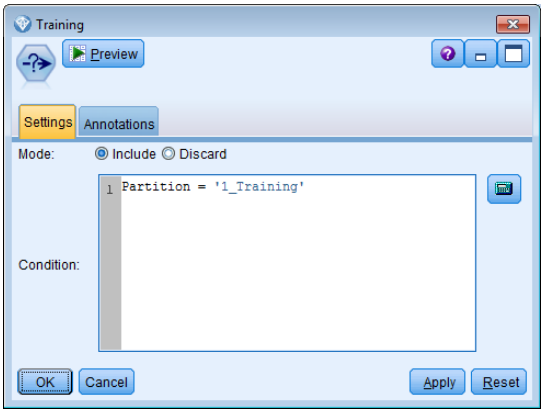
4. Through some trial and error I found that Modeler did not like the names of some of the columns where special characters were used: [,],',;,$,#. Let's add a Filter node form the Field Ops palette and rename these columns.

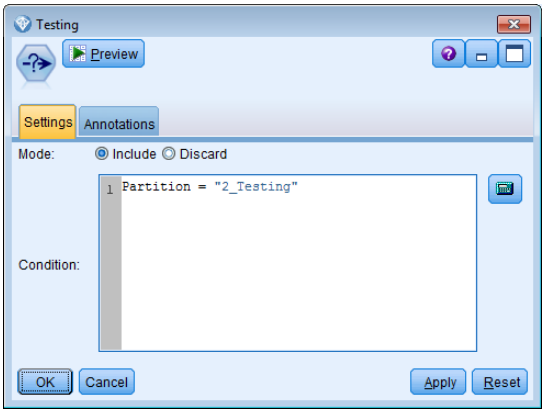


5. With our data prepared, let's partition and split into training and testing
   a. Add a Partition node from the Field Ops Palette
   b. I chose to do 80/20 train/test split for this data.

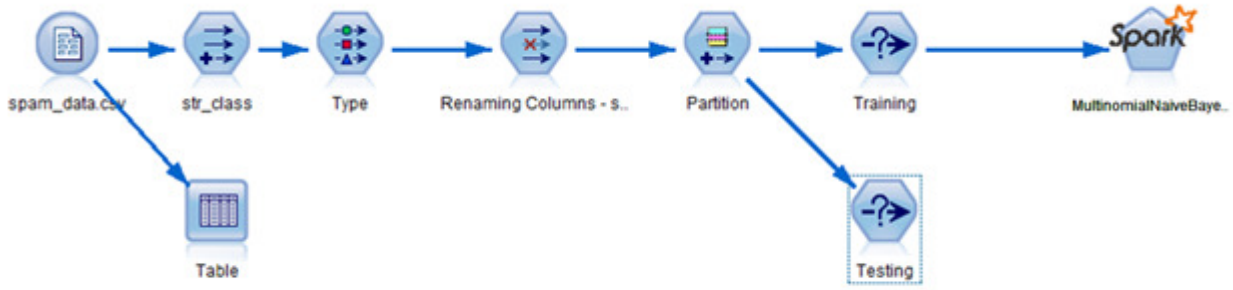


c. This node will create a new column that gives a field to select training and testing from (the values in this column are "1_Training" and "2_Testing")
d. To split the data, add two Select nodes from the Record Ops palette where the formula should match the Nodes below

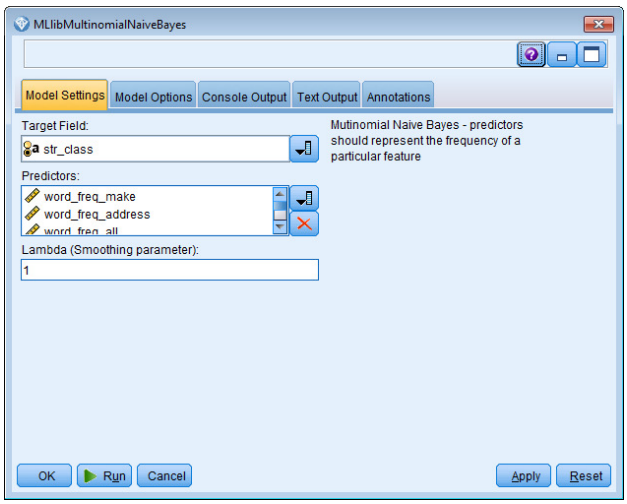




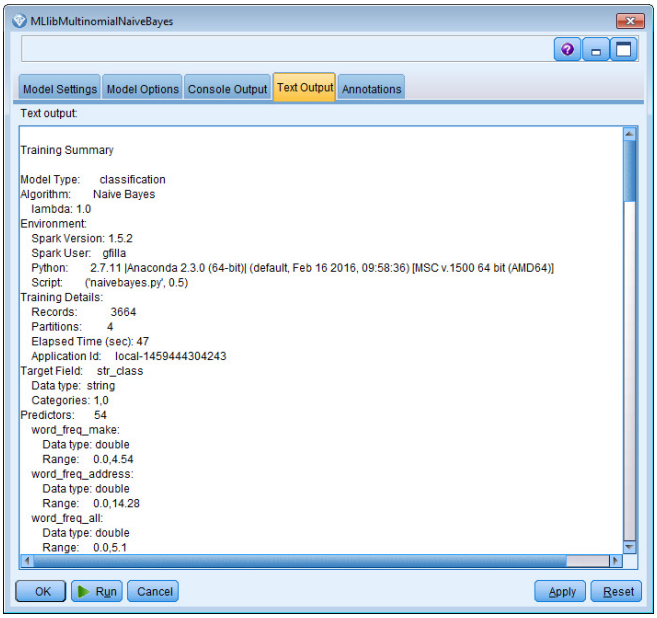
6. Now, let's add our Multinomial Naïve Bayes node so our stream will look like this:



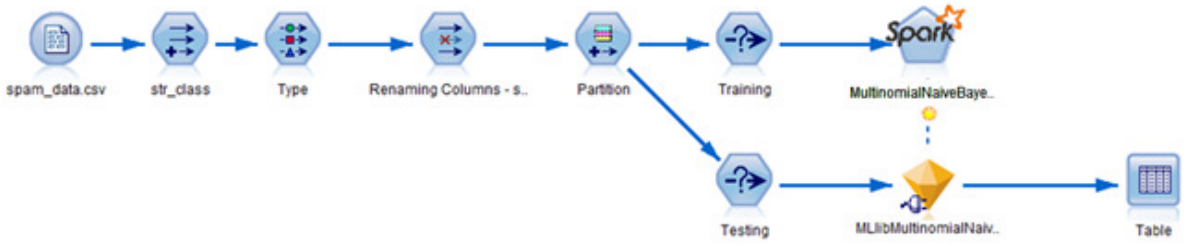
7. Open the Multinomial Naïve Bayes dialog
   a. Select the str_class field for the target
   b. Select all the predictors you want to include for prediction. I included all variables except the 'capital_run_*' variables
   c. Adjust the Lambda parameter as needed; I kept the default of 1.



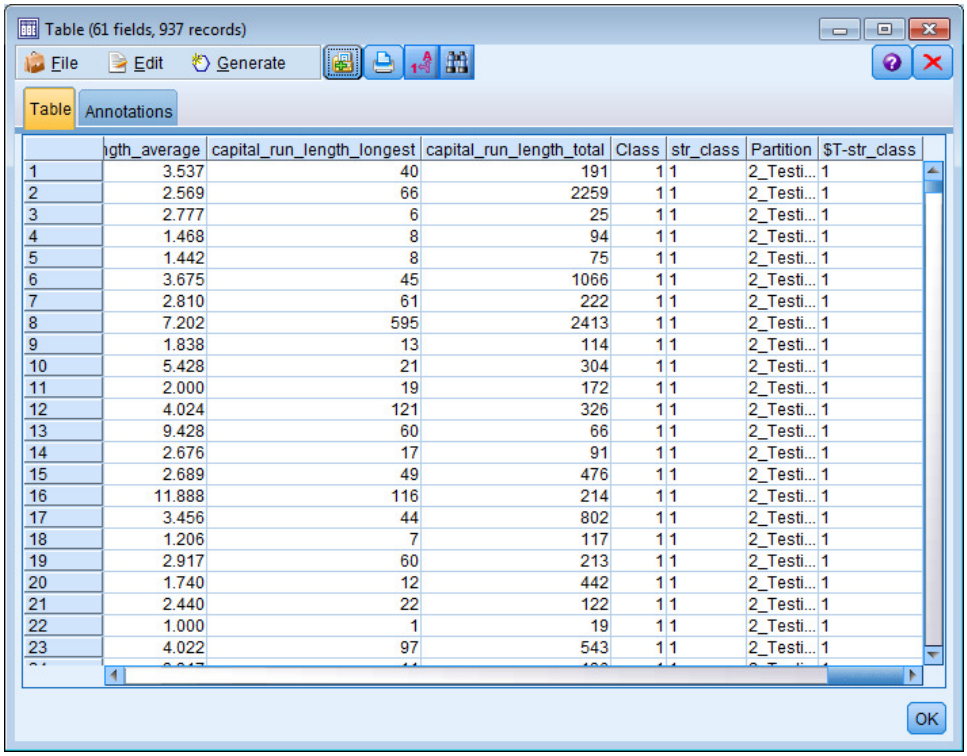
8. Run this stream to create the model.
9. By double clicking the Model Node and clicking on the Text Out tab, you can see a summary of the model training. This may come in handy when looking into details on the model.



10. Now connect the “Testing” Select node to the model nugget create and add a table as the output

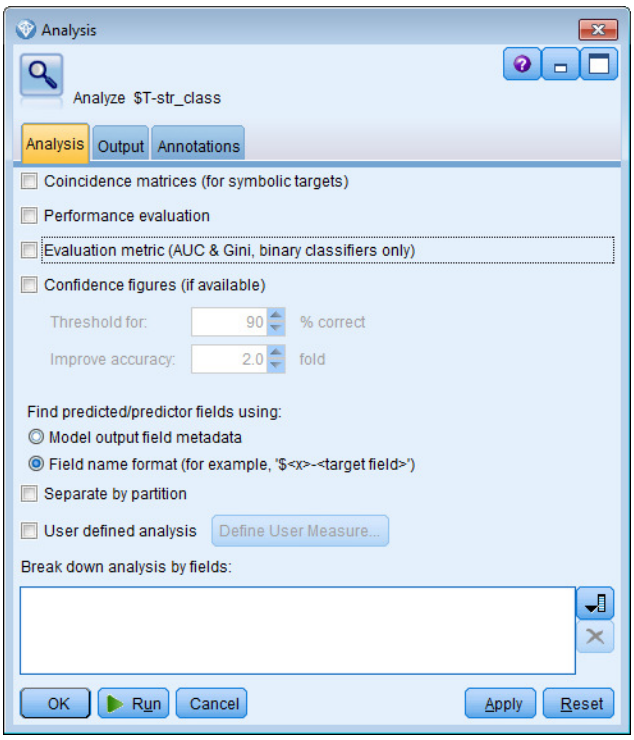


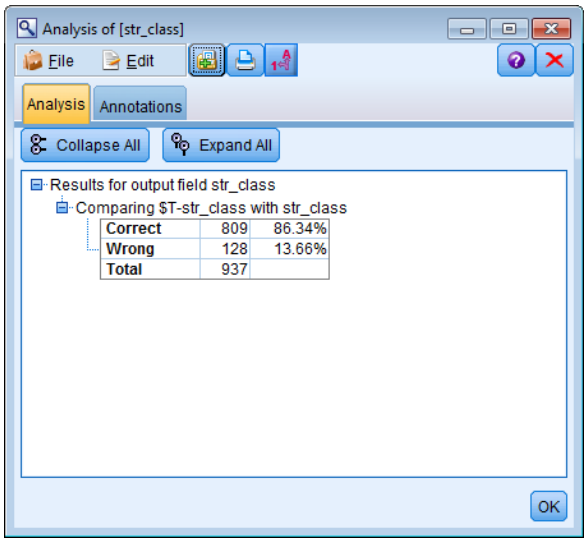
11. Right click the table and Run the stream. The table produced will include the classification for each email:



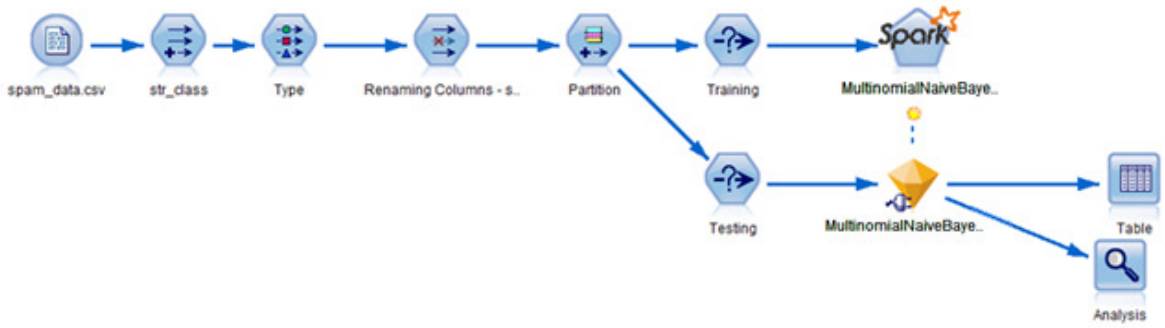
	ngth_average	capital_run_length_longest	capital_run_length_total	Class	str_class	Partition	$T-str_class
1	3.537	40	191	1	1	2_Testi...	1
2	2.569	66	2259	1	1	2_Testi...	1
3	2.777	6	25	1	1	2_Testi...	1
4	1.468	8	94	1	1	2_Testi...	1
5	1.442	8	75	1	1	2_Testi...	1
6	3.675	45	1066	1	1	2_Testi...	1
7	2.810	61	222	1	1	2_Testi...	1
8	7.202	595	2413	1	1	2_Testi...	1
9	1.838	13	114	1	1	2_Testi...	1
10	5.428	21	304	1	1	2_Testi...	1
11	2.000	19	172	1	1	2_Testi...	1
12	4.024	121	326	1	1	2_Testi...	1
13	9.428	60	66	1	1	2_Testi...	1
14	2.676	17	91	1	1	2_Testi...	1
15	2.689	49	476	1	1	2_Testi...	1
16	11.888	116	214	1	1	2_Testi...	1
17	3.456	44	802	1	1	2_Testi...	1
18	1.206	7	117	1	1	2_Testi...	1
19	2.917	60	213	1	1	2_Testi...	1
20	1.740	12	442	1	1	2_Testi...	1
21	2.440	22	122	1	1	2_Testi...	1
22	1.000	1	19	1	1	2_Testi...	1
23	4.022	97	543	1	1	2_Testi...	1

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.
13. This will give us a quick evaluation of our model. For this example we are classifying 86% of the emails correctly.





14. Finally – our full stream should look like this:



Important Links

Learn

- Learn more about SPSS software.
- To learn more about this implementation of K-means take a look at the Spark documentation
- Read about coding a PySpark Extension for SPSS Modeler
- Visit developerWorks Business analytics for more technical analytics resources for developers.

Discuss

- Visit the IBM SPSS Community to share tips and experiences with other IBM SPSS developers.
- Follow developerWorks on Twitter to be among the first to hear about new resources.

References

[1] Apache Spark version 1.5.2 – Mllib Guide: Naïve Bayes (https://spark.apache.org/docs/1.5.2/mllib-naive-bayes.html)

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

[3] Creators: Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304 Donor: George Forman (gforman at nospam hpl.hp.com) 650-857-7835