pitney bowes

Data Science Supervised Learning Demonstration

Loan Default Prediction Using Lending Club Data

Spectrum™ Technology Platform

May 2018



Table of Contents

Introduction 3
Scenario 3
Process Overview 3
Lending Club Data 4
Prepare Data 5
Import the 2007-2015 Dataset 5
Import the 2016 Q1 Dataset 6
Import the 2016 Q2 Dataset 6
Combine Datasets 6
Create a Category Variable Using the Information Extraction Module 7
Prepare Data 8
Configure the Training Options File 8
Train the Model 9
Categorize Text 11
Derive Category Fields from the Output List 12
Create Category Variables 13
Store Data 16
Create Credit Risk Models 18
Logistic Regression Modeling 18
Random Forest Classification Modeling 21
Score New Data Using Java Model Soring 24

Introduction

The purpose of this workbook is to demonstrate the functionality of the Spectrum™ Technology Platform Data Science Solution in Enterprise Designer. More specifically, it demonstrates how to predict loan defaults using the Machine Learning Module and then how to score new data using the Analytics Scoring Module. It works in conjunction with the provided dataflows, training data, and test data. You can create your own dataflows by following the step-by-step instructions in this workbook, or you can use the included dataflows as a reference to confirm what the individual completed stages and output should look like. If you prefer the latter option and are running Spectrum™ Technology Platform 12.2, use the following dataflows:

- **LendingClub_2007_2016Q1Q2_v122_SingleCategorizer.df**—creates Logistic Regression and Random Forest Classification models
- **LendingClub_2016Q3_v122_SingleCategorizer_Scoring.df**—scores those models using newer data

If you are running Spectrum™ Technology Platform 12.1, use the similarly named dataflows but with “v121” instead of “v122”. If you are creating your own dataflows while following these instructions, you will first create a job in the Prepare Data section and will continue in that job until the Score New Data Using Java Model Scoring section, where you will create a scoring dataflow.

Also included in the zip file is a dataflow for using multiple Text Categorizer stages. Note that at this time we do not provide a scoring dataflow for multiple categorizers, and the scoring dataflow for a single categorizer is not compatible with multiple categorizers.

Stages covered in this workbook include the following; click a stage to access its webhelp for more information:

- Sources: Read from File
- Sinks: Write to File
- Control stages: Stream Combiner, Broadcaster, Splitter, and Transformer
- Analytics Scoring stages: Write to Miner Dataset
- Information Extraction stages: Text Categorizer
- Machine Learning stages: Logistic Regression and Random Forest Classification

Scenario

You are a data analyst at Lending Club and have been asked to produce a dataset that you will subsequently use to perform credit risk modeling and apply machine learning techniques to better predict whether potential borrowers will default on a loan.

Process Overview

We complete the following steps in this workbook:

Prepare Data

Data preparation is the construction of datasets for the purpose of carrying out customer analytics. In practice, data preparation may be undertaken in different ways, depending on the nature and location of stored data and the facilities available to build customer datasets.

Combine Datasets

Working with data in this manner often requires multiple analyses that use different raw data. A repeat analysis might be based on a different time period or a different group of customers. In such cases, the structure of the dataset required for the analysis might be exactly the same as the original, but it may need to be updated or "refreshed" with new raw data. For example, a dataset built using June 2016 data may require updating with July 2016 data when it becomes available. The amount of effort required to carry out an update will vary depending on the processes used. In our project, we use multiple data sets that together span several years.

When working with data from multiple sources or with multiple vintages, it is sometimes necessary to combine the files to form the strongest, most comprehensive dataset. In our job, we use a Stream Combiner to bring together all the data and save it as one FTR file using the Write to Miner Dataset stage. This is particularly useful if you are licensed for Spectrum Miner™.

Create a Category Variable Using the Information Extraction Module

We follow several steps to create a category variable: prepare data, configure options, train the model, categorize the text, derive category fields from an output list, and, finally, create the category variable.

Store Data

After creating variables you should save the new dataset. We create a CSV file of this data and export the metadata to an XML file.

Create Credit Risk Models

The final step in this dataflow is to create models that determine credit risk. Specifically, we will apply Logistic Regression and Random Forest Classification models to predict whether potential borrowers will default on their loans. Logistic Regression enables you to perform machine learning by creating models from datasets that use binary objectives with input variables. Random Forest Classification also enables you to perform machine learning, but in this case you create models from datasets that use binary or multinomial objectives with input variables.

Lending Club Data

Lending Club is a peer-to-peer online lending platform and the world's largest marketplace connecting borrowers and investors. Consumers and small-business owners lower the cost of their credit and enjoy a better experience than with traditional bank lending, while investors earn attractive, risk-adjusted returns. The entire process is online, using technology to lower the cost of credit and pass the savings back in the form of lower rates for borrowers and solid returns for investors.

At a high level, the lending process looks like this:

1. Customers interested in a loan complete a simple application at LendingClub.com.
2. Lending Club leverages online data and technology to quickly assess risk, determine a credit rating, and assign appropriate interest rates.
3. Qualified applicants receive offers in just minutes and can evaluate loan options with no impact to their credit score.
4. Investors ranging from individuals to institutions select loans in which to invest and can earn monthly returns.

Data Download

Lending Club makes their loan data available, accounting for privacy, of course. For this demonstration, we want to use loan data for years 2007-2015 and the first three quarters of 2016. Kaggle provides a dataset for all Lending Club loans issued from 2007 through 2015; you can download that data at https://www.kaggle.com/wendykan/lending-club-loan-data. Lending Club updates the standard dataset every quarter; you can download the 2016 files at https://www.lendingclub.com/info/download-data.action. Specifically, you will download loan.csv from Kaggle and LoanStats_2016Q1.csv, LoanStats_2016Q2.csv, and LoanStats_2016Q3.csv from Lending Club.

Data Specifics

All the data is stored in comma-separated values (CSV) files that contain complete loan data, including loan status (Current, Late, Fully Paid, etc.) and latest payment information. Each loan is identified by a unique key in the *Id* field. There are several fields populated with NA values and several fields that were populated after the loan was issued. Since we are modeling whether a loan should be offered or not, we are only concerned with the information provided to Lending Club when the loan was requested. Therefore, we will review the data and select only the variables that help guide that decision.

Prepare Data

In these instructions, we assume that the Spectrum™ Technology Platform server and Enterprise Designer are both installed on your local machine. Additionally, we assume that you have downloaded the aforementioned Lending Club data.

Import the 2007-2015 Dataset

As discussed in the scenario, the first task is importing the complete loan data for all loans issued from 2007 through 2015.

1. Open Enterprise Designer.
2. Select File > New > Dataflow > Job.
3. Drag a Read from File stage onto the canvas and double-click that stage.
4. In the **File name** field, locate and specify the file *loan.csv*.
5. In the **Record type** field, choose Delimited.
6. In the **Field separator** field, select Comma (,).
7. In the **Text qualifier** field, select Double quote (“)
8. In the **Record separator** field, select Unix (U+000A).

9. Check **First row is header record**.
10. Uncheck **Treat records with fewer fields than defined as malformed** and click OK.
11. Select Edit > Job options and check **Do not terminate the job on a malformed record** and click OK.
12. Select Edit > Type Conversion Options and check **Override system default options with the following values**. Select **Initialize the field using default values**, uncheck all **Fail Null** boxes, and then click OK.
13. Save the job.

Import the 2016 Q1 Dataset

1. The data in *LoanStats_2016Q1.csv* contains notes on the first row and the last two rows; you must remove these rows before importing the data in Enterprise Designer.
2. Repeat steps 3-10 above to import the 2016 Q1 data.
3. Save the job.

Import the 2016 Q2 Dataset

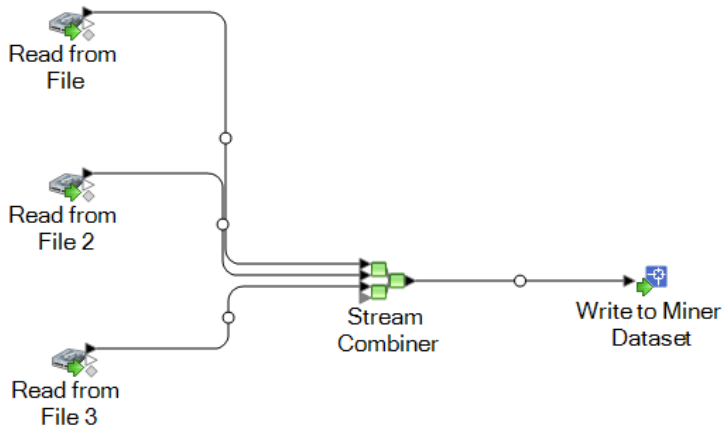
1. The data in *LoanStats_2016Q2.csv* contains notes on the first row and the last two rows; you must remove these rows before importing the data in Enterprise Designer.
2. Repeat steps 3-10 again to import the 2016 Q2 data.
3. Save the job.

Combine Datasets

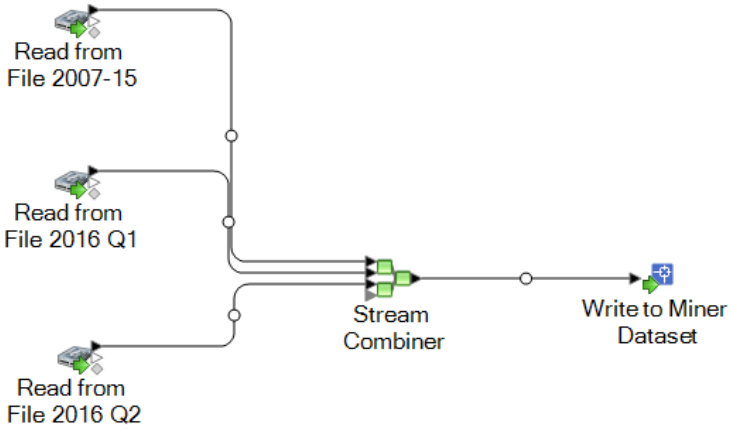
You have just imported data from each of your two initial sources. Now you need to combine all three sources into one dataset.

1. Drag a Stream Combiner stage onto the canvas.
2. Click the solid black triangle on the right side of the Read from File stage (the output port) and drag it to the gray triangle on the left side of the Stream Combiner stage to create a channel connecting the two stages.
3. Repeat step 2 for the remaining datasets.
4. If you are licensed for Spectrum Miner™, you can thoroughly explore this data by creating a Miner dataset following steps a-e below. If you are not licensed for Spectrum Miner™, you can add a Write to Null stage here or, alternatively, add a Write to File stage to capture the combined data.
    a. Drag a Write to Miner Dataset stage onto the canvas.
    b. Double-click the stage and in the **File Name** field enter the path and name of your new Miner dataset.
    c. Click the Fields tab and add the following fields:
        - id
        - member_id
        - loan_status
        - desc
        - title
    d. Click OK.
    e. Save the job.

Your dataflow should look like this:



You can edit the labels for the input stages to reflect their contents:



Create a Category Variable Using the Information Extraction Module

In the combined data file you just created, we include a field called “desc”, which stores the reason for the loan. In this task, we assign custom categories to this field using the Information Extraction Module Text Categorizer stage. The first step will be to "train" a model to "learn" the categories. After this, we can use that model in the Text Categorizer stage to categorize the “desc” field in the combined data file.

Spectrum™ Technology Platform uses administration utility commands to manage text categorization models. For a description of these commands, search Administration Utility Commands in the webhelp for the Information Extraction Guide.

Prepare Data

The first step in using text categorization is preparing the training file and the test file. These files must use UFT-8 encoding, and the data must be structured with tab-separated values in two columns, where the first column contains the category name (for example, " isStable" or "isThankful") and the second column contains the value for each category, as displayed in the example below. Note that some records must be labeled as not meeting any category.

isStable	I have been with my job now for over 9 years and it has been very stable. My plans are to pay off my credit card debt and in turn raise my credit rating.
isThankful	Thank you sooooo much for your help...this will make life a little bit easier! Every little bit helps, i will finally be able to sleep at night

Configure the Training Options File

This step involves creating a training options file that contains information about your model and the options to be applied to train the model. In this demonstration, we use **training.xml** as the training options file; be sure to edit this file to include the correct locations for inputFilePath and testFilePath.

This file must be in XML format with UFT-8 encoding and must include the provided header fields as well as the required training features.

Training Options File Header

The header in the training options file includes details of the model, its type, and the paths of the test and input files:

- modelName: Name of the model
- modelType: The type of model (for our purposes, text categorization)
- modelDescription: Description of the model
- inputFilePath: Location of the input file used for training the model
- testFilePath: Location of the test file

Training Features

Training features are used to create new categories. In this demonstration, we use the following training features:

Linguistic Features

A linguistic feature specifies language properties. In particular, we use Stemming, which reduces words to their stem, or root. For example, "insurer", "insured", and "insures" can all be reduced to the root "insure".

Keyword Features

Keyword features define lists of keywords.

- IgnoreWords—Also known as stop words, this feature filters out common words that have no effect on categorization, such as “the”, “and”, and “but”. These words should be separated only by a comma, not by spaces. You can also use the Append key with this feature, which, when set to “True”, will be added to the existing list of stop words.

- CategoryKeywords—This feature identifies a category for a list of keywords belonging to multiple custom lists. For example, Weekdays in a CategoryKeywords list contains keywords as Monday, Tuesday, Wednesday, Thursday, and Friday.
- KeyWords—This feature searches for words that you have specified as belonging to a custom list, such as “DaysOfWeek” or “Month”.

Lexical features

Lexical features specify the lexeme properties, which assist in the testing and training of data; our demonstration uses Ngram, which searches for a portion of a longer string, with “n” representing the number of words to look for. For example, if you are looking for the phrase “to be or not to be”, you might search for a unigram of “to” or “be”, or a bigram of “to be” or “or not”, or a trigram of “to be or” or “not to be”.

Train the Model

After creating a training options file, the next step is to train your model to discover potentially predictive relationships by applying machine learning methods to learn automatic classification rules. Spectrum™ Technology Platform performs this function using the command-line interface (CLI) command “iemodel train”, which is documented in the Administration Utility Commands section of the Information Extraction Guide.

1. Go to the Spectrum™ Technology Platform Welcome Page. If the server was installed on a computer named "myspectrumplatform" and it is using the default HTTP port, you would go to: http://myspectrumplatform:8080. If the Spectrum™ Technology Platform was installed locally, you can go to http://localhost:8080/dcg/.
2. Click Platform Client Tools.
3. Click Command Line.
4. Under Administration Utility, click **Download** to download the zip file to the computer where you want to use the Administration Utility.
5. Extract the contents of the zip file somewhere on that same computer.
6. To launch the command line interface, do one of the following:
   If you are running the server on a Unix or Linux system, execute **cli.sh**.
   If you are running the server on a Windows system, execute **cli.cmd**.
7. If the Spectrum™ Technology Platform was installed locally, you can connect to the server by typing this command:

   ```
   connect --h localhost:8080 --u admin --p admin
   ```

   If the Spectrum™ Technology Platform was installed on a computer named “myspectrumplatform” and is using the default HTTP port, you would type:

   ```
   connect --h myspectrumplatform:8080 --u admin --p admin
   ```

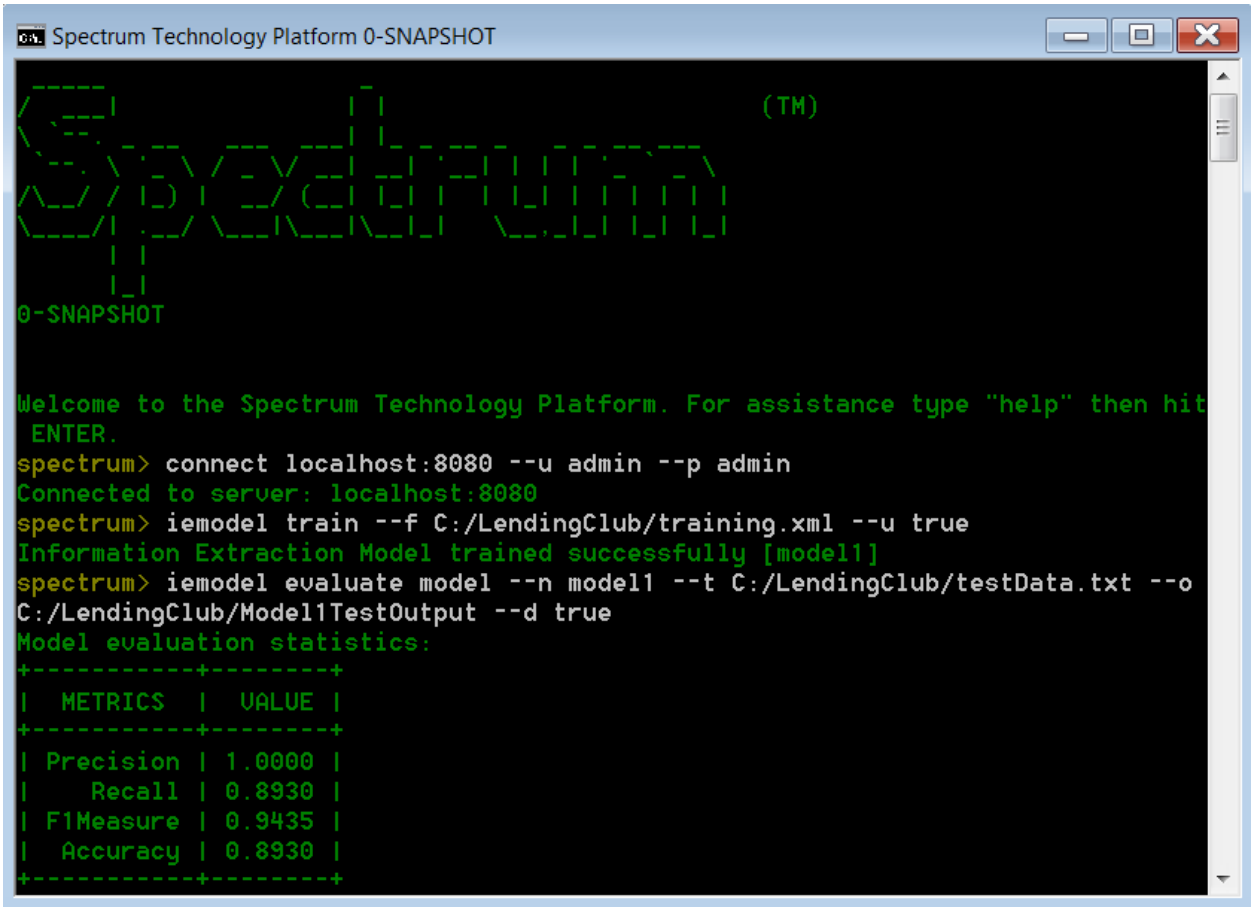
8. Once you are connected, you can run the following command, changing the path as necessary for your environment. This command calls the training options file we just created, which points to the input file and applies options. It also allows you to specify whether to overwrite any existing model with the same name with the “u” command.

   ```
   iemodel train --f C:/LendingClub/training.xml --u true
   ```

After you have trained your model, you can use this command to evaluate it. It allows you to test your model after training to verify that the options file is correct and that categories are being assigned as expected. Results include evaluation results such as category counts and a confusion matrix. The “t” command, which is required, calls your test file. The “o” command, which is optional, calls your output file name. If you do not add the “o” command, the output will not be stored. Finally, the “d” command, also optional, allows you to specify whether to display a table with detailed analysis in the results.

```
iemodel evaluate model --n model1 --t C:/LendingClub/testData.txt --o
C:/LendingClub/Model1TestOutput --d true
```

A successful use of these commands looks similar to this:



Alternatively, you can train and evaluate your model using a single command. This combination of the two commands calls the training options file, allows you to specify whether to overwrite any existing model with the same name, calls the output file, and allows you to specify whether to display a table with detailed analysis, respectively.

```
iemodel trainAndevaluate model --f C:/LendingClub/training.xml --u true
--o C:/LendingClub/Model1TestOutput --d true
```

The output from a successful use of these commands looks similar to this:

Model1TestOutput

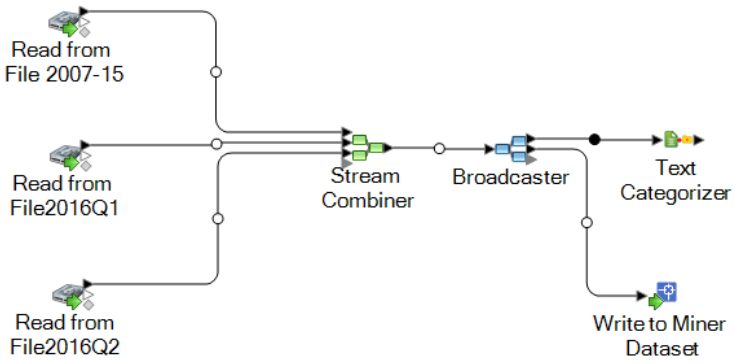
```
1  METRICS,VALUE,DESCRIPTION
2  Precision,1.0000
3  Recall,0.8930
4  F1Measure,0.9435
5  Accuracy,0.8930
6
7  Model evaluation results:
8  Category Name,Input Count,Mismatch Count,Match Count,
9  isCollegeRelated,200,0,200
10 isThankful,198,39,159
11 NoCategory,400,30,370
12 isStable,183,36,147
13 All,981,105,876
14
15 Confusion Matrix:
16 Confusion Matrix, ,Predicted,
17 ,,POSITIVE,NEGATIVE
18 ACTUAL,TRUE,876,0
19 ,FALSE,0,105
```

After the model is trained, it is ready to use in categorization.

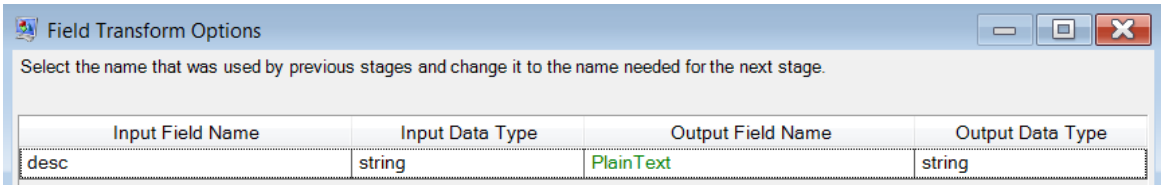
Categorize Text

1. Drag a Broadcaster stage onto the canvas and insert it in between the Stream Combiner stage and the Write to Miner Dataset stage.
2. Drag a Text Categorizer stage onto the canvas and connect it to the Broadcaster stage.

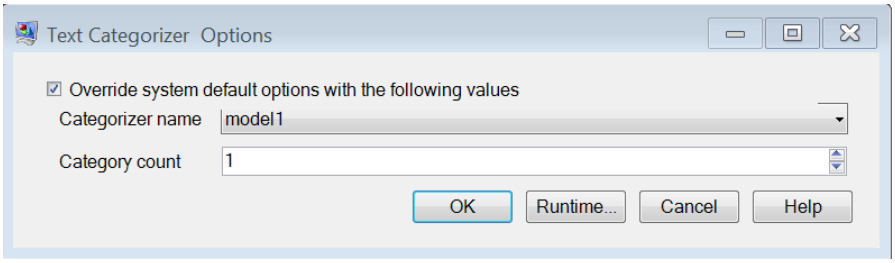
Your dataflow should look like this:



3. Double-click the channel between Broadcaster and the Text Categorizer to access the Field Transform Options window.
4. Select **desc** for the Input Field Name and click OK to close the window.



5. Double-click the Text Categorizer stage to access the Text Categorizer Options window.
6. Check **Override system default options with the following values**.
7. Select "model1" (the trained model specified on line 3 of training.xml) in the **Categorizer name** field and select "1" in the **Category count** field, which is the highest matching level of category (the closest match).

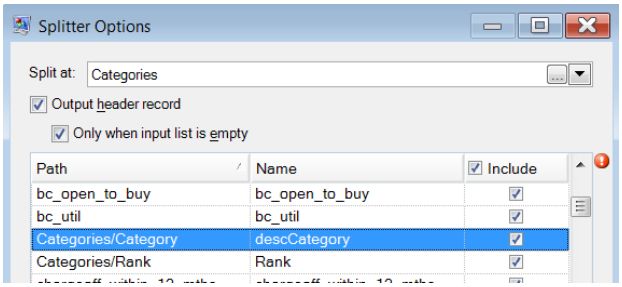


8. Click OK and save the job.

Derive Category Fields from the Output List

The output from the Text Categorizer stage is a list. To derive useful information from this list we need to use the Splitter stage.

1. Drag a Splitter stage onto the canvas and connect it to the Text Categorizer stage.
2. Double-click the Splitter stage to access the Splitter Options window.
3. Select "Categories" in the **Split at** field.
4. Check **Output header record** and **Only when input list is empty**.
5. Find the row with **Path** name "Categories/category" and change the name to "descCategory".



6. Check the **Include** box and then click OK.

Create Category Variables

The derivation of new fields within Enterprise Designer is an important part of data preparation and data preprocessing activities. When an existing field is not in an appropriate form for analysis, a new field may be derived that transforms the existing field's values into a more suitable form. Or, you can derive a result field that contains a value calculated from one or more other fields in the file.

We will first encode our dependent variable from the loan_status field, with value "Charged Off" being equal to 1 if it enters default; otherwise it will be equal to 0. This will allow us to fit logistic regression and categorical models to the data. After that we will use the Transformer stage to perform data transformations required on predictive variables.

The Transformer stage has predefined transforms that perform a variety of common data transformations. If the predefined transforms do not meet our needs, we can write a custom transform script using Groovy. Click here for more information on creating custom transforms and here for more information on using Groovy scripts.

1. Create a dependent variable

a. Drag the Transformer stage onto the canvas and open the stage.
b. Click Add. Under the **General** heading, click "Custom".
c. Enter a name for the transform you are creating in the **Custom transform name** field. The name must be unique.
d. Click **Script Editor** and then click the green cross in the Output Fields pane to access the Add Custom Field window. Click Add.
e. Enter "Loan_Default" as the **Field name** and select "Integer" as the data type. Then click OK twice and enter the following code in the script editor pane:

```
if (data['loan_status'].equals('Charged Off')){ data['Loan_Default'] =1;}
else data['Loan_Default'] =0;
```

f. Click Close twice, then OK.

2. Work with "date" format: create independent variable "time_history"

Our data includes the "issue_d" field, which contains the month the loan was funded. It also contains the "earliest_cr_line" field, which is the month the borrower's earliest reported credit line was opened.

We want to create a variable with information regarding how long from the date of application the first credit account was opened until the loan was issued. We do this first by converting the dates into a workable format and then subtracting the difference. You will notice that the day is not given in the dates from Lending Club; we can get only an approximate value for each observation.

The **issue_d** field has a format of "16-Mar" for 2016 Q1 data but a format of "Jun-16" for 2016 Q2 data. The **earliest_cr_line** field uses both date formats in one dataset:

When converting strings such as “3-Jul” to date format, we need to concatenate a “0” at the beginning of the string. We will use the Transformer stage to do this. Repeat e-g from Step 1 above to add new variables “issue_d_new”, “earliest_cr_line_new” and “time_history”. Enter the following code in the script editor pane:

```
if (!data['issue_d'].equals(null)){
      data['issue_d1'] =substring(data['issue_d'] , 0, 1);
      data['issue_d2'] =substring(data['issue_d'] , 0, 2);

      if (data['issue_d2'].isNumber())
      {     data['issue_d_new'] =getDateFromString(data['issue_d'],
"YY-MMM");}
      else if (data['issue_d1'].isNumber())
      {
            data['issue_d'] = '0'+data['issue_d'];
            data['issue_d_new']
=getDateFromString(data['earliest_cr_line'], "YY-MMM");
      }
      else { data['issue_d_new'] =getDateFromString(data['issue_d'],
"MMM-YY");}

      if (!data['earliest_cr_line'].equals(null)){

            data['erliest_cr_line1']
=substring(data['earliest_cr_line'] , 0, 1);
            data['erliest_cr_line2']
=substring(data['earliest_cr_line'] , 0, 2);

            if (data['erliest_cr_line2'].isNumber())
            {     data['earliest_cr_line_new']
=getDateFromString(data['earliest_cr_line'], "YY-MMM");}
            else if (data['erliest_cr_line1'].isNumber())
            {
                  data['earliest_cr_line'] =
'0'+data['earliest_cr_line'];
                  data['earliest_cr_line_new']
=getDateFromString(data['earliest_cr_line'], "YY-MMM");
            }
            else { data['earliest_cr_line_new']
=getDateFromString(data['earliest_cr_line'], "MMM-YY");}

            data['time_history'] =dateDifference(
data['earliest_cr_line_new'] , data['issue_d_new'] ,4);
      }
}
```

3. Work with string: parse a field and convert it to double

We need to transform some of the columns that have text in the fields such as “revol_util” (revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit), which contains “%” for the percentage. Enterprise Designer reads this as a string and the calculations on this field will not work as a numeric. We need to replace the percent sign with an empty space and convert a string to numeric type.

We will use the same Transformer stage, repeating e-g from Step 1 above to create a new variable called “RevolUtilizRate.” Enter the following code in the script editor pane:

```
if (!data['revol_util'].equals(null) && data['revol_util']?.trim()){
 String revol_util = data['revol_util'].replaceAll("%", "");
 data['RevolUtilizRate'] = Double.parseDouble(revol_util);
}
```

4. Work with string: use of missing data

Valuable information can be ascertained from null values. Below are three variables we extract using null data:

- Whether the borrower left their employer’s name blank.
- Whether they wrote a description of what they are going to be using the funds for. This is very predictive of default.
- How long it has been since they defaulted. Most of these variables will be NA because the majority of borrowers have not defaulted, but we are definitely concerned with whether the borrower has defaulted in the past.

Using the same Transformer stage, repeat steps e-g from Step 1 above to create new variables “employer_listed”, “empty_desc” and “delinq_ever”. Enter the following code in the script editor pane:

```
if (data['emp_title'].equals(null)){ data['employer_listed'] =0;}
else data['employer_listed'] =1;
if (data['desc'].equals(null)){ data['empty_desc'] =1;}
else data['empty_desc'] =0;
if (data['mths_since_last_delinq'].equals(null)){ data['delinq_ever']
=0;}
else data['delinq_ever'] =1;
```

5. Work with string: parse a field and convert it to integer

The emp_length variable has a string format (e.g., “2 years”); we need to parse integers from this string. We will use the same Transformer stage, repeating e-g from Step 1 to create new integers “emp_length_na” and “emp_length_new”. Enter the following code in the script editor pane:

```
if (data['emp_length'].equals('n/a')){ data['emp_length_na'] =1;}
else data['emp_length_na'] =0;

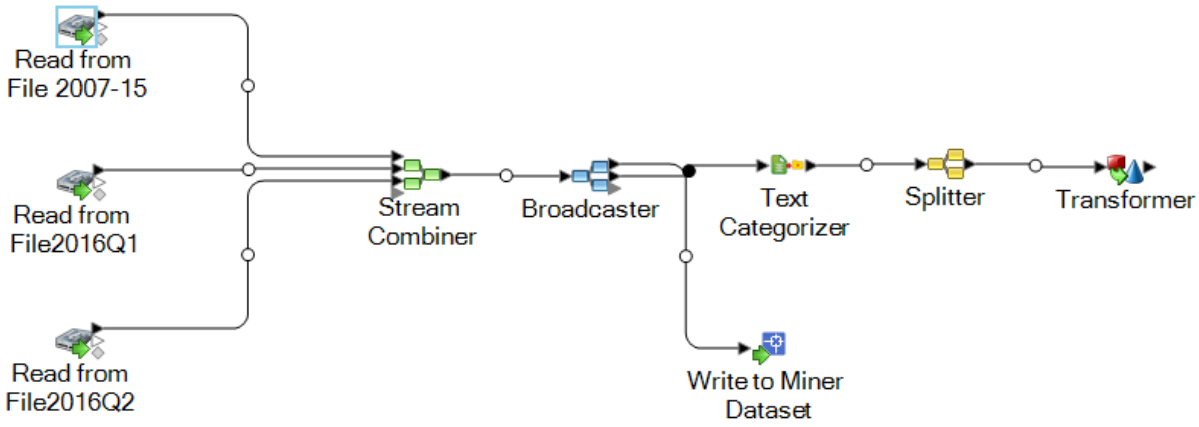
if (data['emp_length'].equals(null) || data['emp_length'].equals('< 1
year')||data['emp_length'].equals('n/a')){ data['emp_length'] ="0";}
String  emp_length = data['emp_length'].findAll(/\d+/)[0];
data['emp_length_new'] = emp_length.toInteger();
```

6. Work with string: cleaning

Here we want to alter values in the “home_ownership” field. There are five categories; one is called “NONE” and another is called “OTHER”. We need to combine these two categories into the “OTHER” category. Enter the following code in the script editor pane:

```
if (data['home_ownership'].equals('NONE')){ data['home_ownership']
="OTHER";}
```

Your dataflow should look like this:



Store Data

Now that we have created new variables, the next step is to save our data. We want to select only the relevant fields for the analysis.

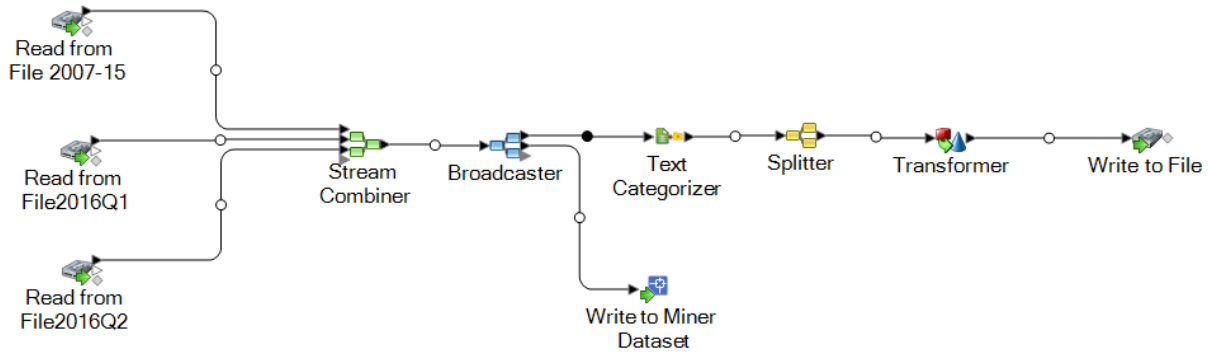
1. Drag a Write to File stage onto the canvas.
2. Connect the Splitter stage and the Write to File stage.
3. Double-click the Write to File stage.
4. In the **File name** field, save the data as “loan_data_2007_2016Q1Q2_output.csv”.
5. In the **Record type** field, choose Delimited.
6. In the **Field separator** field, select Comma (,).
7. In the **Text qualifier** field, select Double quote (“).
8. In the **Record separator** field, select Unix (U+000A).
9. Check **First row is header record**.
10. Click the Fields tab.
11. Click **Quick Add**, then **Deselect All**.

12. Select the following variables, ensuring the proper Input Data Type:

Input Field Name	Input Data Type
annual_inc	string
delinq_2yrs	string
dti	string
home_ownership	string
inq_last_6mths	string
loan_amnt	string
Loan_Default	integer
open_acc	string
pub_rec	string
purpose	string
time_history	integer
total_acc	string
verification_status	string
delinq_ever	integer
emp_length_na	integer
emp_length_new	integer
employer_listed	integer
empty_desc	integer
RevolUtilizRate	double
descCategory	string
id	string
member_id	string

13. Export the metadata for future use:
    - Click the File Properties tab.
    - Click the Export button.
    - Save the metadata xml file and name it “loan_data_2007_2016Q1Q2_output.xml”.
    - Click Save and OK twice.
14. Click OK and save the job.

    Your dataflow should look like this:



15. Click the Run button ( ) to run your dataflow. The Execution Details window appears and shows the status of the job.
16. Click Refresh. Once the status shows Succeeded, click Close.

Open the output file you specified in the Write to File stage. You will see the records you selected.

annual_inc	delinq_2yrs	dti	home_ownership	inq_last_6mths	loan_amnt	Loan_Default	open_acc	pub_rec	purpose	time_history	total_acc	verification_status	delinq_ever	emp_length_na	emp_length_new	employer_listed	empty_desc	RevolUtilizRate
88000	0	22.98	MORTGAGE	0	30000	0	15	0	debt_consolidation	4627	36	Source Verified	0	0	2	1	1	43.5
135000	0	20.2	OWN	0	24000	0	14	0	credit_card	4930	19	Verified	1	0	1	1	1	67.9
55000	0	33.71	RENT	0	12500	0	11	0	debt_consolidation	3074	18	Verified	0	0	5	1	1	73.3
138000	0	12.44	RENT	1	25000	0	9	1	debt_consolidation	7518	24	Not Verified	0	0	10	1	1	37.1
40000	0	36.18	RENT	0	16800	0	14	1	debt_consolidation	6634	35	Source Verified	0	0	10	1	1	41.6
134000	0	26.33	MORTGAGE	1	22000	0	20	0	debt_consolidation	16010	34	Verified	0	0	10	1	1	67
30000	0	40.76	RENT	1	12000	0	29	0	debt_consolidation	7334	43	Verified	0	0	10	1	1	16.7

Create Credit Risk Models

Now we will apply Logistic Regression and Random Forest Classification models to predict whether potential borrowers will default on their loans. As you will see in the instructions below, one of the two Logistic Regression stages and one of the two Random Forest Classification stages contain input predictors from a Text Categorizer stage. What we are doing is fitting models with and without the new fields derived from the Text Categorizer stages to determine whether using data in the open text fields improves prediction.

Logistic Regression Modeling

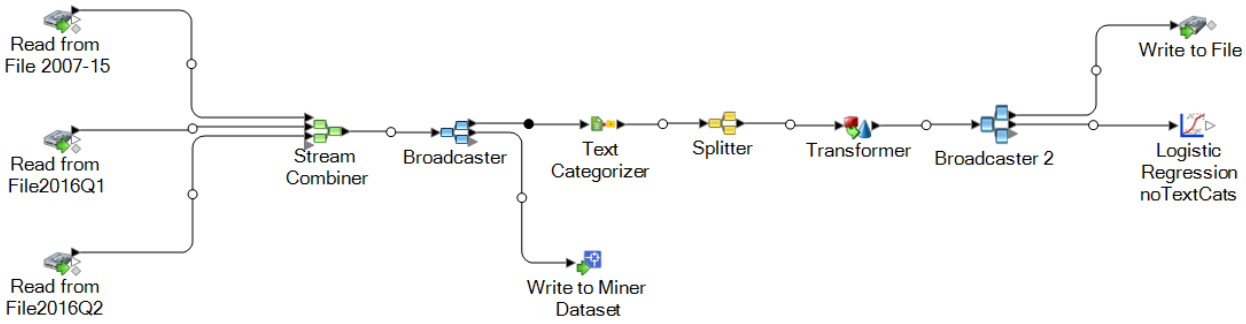
1. Drag a Broadcaster stage onto canvas and connect it to the Transformer stage.
2. Drag a Logistic Regression stage onto the canvas and connect it to the Broadcaster stage.
3. Double-click the Logistic Regression stage; the Model Properties tab appears.
4. In the **Model name** field, specify the model named “LendingClub LR oneiem notused v12_2” (or “LendingClub LR oneiem notused v12_1” if you are running this demonstration on Spectrum™ Technology Platform 12.1).
5. Check **Overwrite** to overwrite the existing model.
6. In the **Objective field**, select Loan_Default.
7. Select the following variables, ensuring the proper Model Data Type:

Input Fields	Type	Model Data Type	Include
Loan_Default	**integer**	**categorical**	☑
time_history	integer	numeric	☑
RevolUtilizRate	double	numeric	☑
employer_listed	integer	categorical	☑
empty_desc	integer	categorical	☑
delinq_ever	integer	categorical	☑
emp_length_na	integer	categorical	☑
emp_length_new	integer	numeric	☑
loan_amnt	string	numeric	☑
home_ownership	string	categorical	☑
annual_inc	string	numeric	☑
verification_status	string	categorical	☑
purpose	string	categorical	☑
dti	string	numeric	☑
delinq_2yrs	string	categorical	☑
inq_last_6mths	string	categorical	☑
open_acc	string	numeric	☑
pub_rec	string	categorical	☑
total_acc	string	categorical	☑

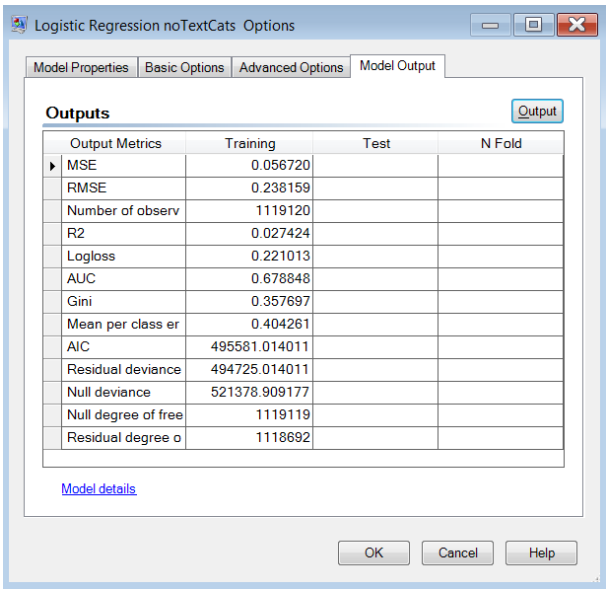
8. Continue to the Basic Options tab.
   a. Leave **Standardize input fields** checked to standardize the numeric columns to have zero mean and unit variance.
   b. Leave **Score input data** unchecked.
   c. Check **Impute means** for missing data fields.
   d. Leave the value 100 as the **Percentage for training data**.
9. Continue to the Advanced Options tab.
   a. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
   b. Check the **Compute p values** checkbox to calculate p values for the parameter estimates.

c. Deselect **Seed for N fold.**
d. Click OK and save the job.

Your dataflow should look like this:



10. Click the Run button to run your dataflow. The Execution Details window appears and shows the status of the job.
11. Click Refresh. Once the status shows Succeeded, click Close.
12. Double-click the Logistic Regression stage, then click the Model Output tab, followed by the Output button. A limited version of the resulting model metrics appears:

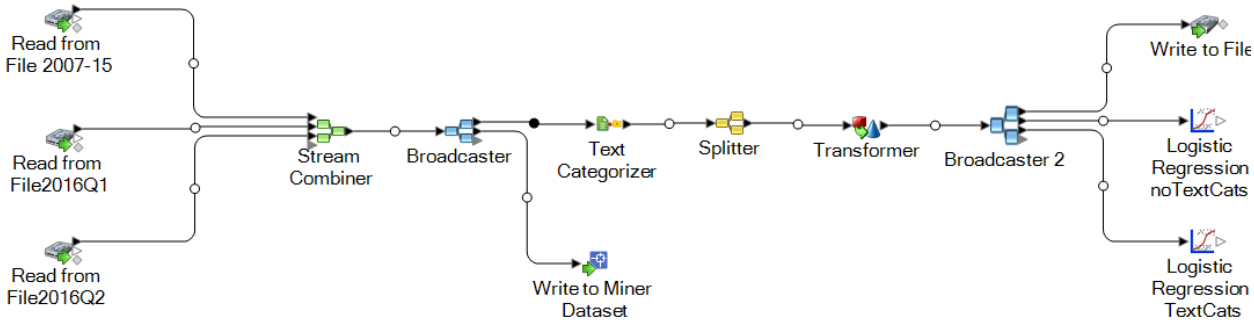


Output Metrics	Training	Test	N Fold
MSE	0.056720		
RMSE	0.238159		
Number of observ	1119120		
R2	0.027424		
Logloss	0.221013		
AUC	0.678848		
Gini	0.357697		
Mean per class er	0.404261		
AIC	495581.014011		
Residual deviance	494725.014011		
Null deviance	521378.909177		
Null degree of free	1119119		
Residual degree o	1118692		

13. Click the Model details link; it will lead you to the Machine Learning Model Management page, where you can see all the metrics for the model you just created. You may need to log on to this page using a provided user name and password.
14. Repeat steps 2-9 to create a second Logistic Regression stage. This time we want to use an input predictor from the Text Categorizer stage.
    a. In the **Model name** field, specify the model named "LendingClub LR oneiem used v12_2" (or "LendingClub LR oneiem used v12_1" if you are running this demonstration on Spectrum™ Technology Platform 12.1).
    b. Select the following variables, ensuring the proper Model Data Type. Note the addition of the descCategory field.

Input Fields	Type	Model Data Type	Include
Loan_Default	**integer**	**categorical**	☑
time_history	integer	numeric	☑
RevolUtilizRate	double	numeric	☑
employer_listed	integer	categorical	☑
empty_desc	integer	categorical	☑
delinq_ever	integer	categorical	☑
emp_length_na	integer	categorical	☑
emp_length_new	integer	numeric	☑
descCategory	string	categorical	☑
loan_amnt	string	numeric	☑
home_ownership	string	categorical	☑
annual_inc	string	numeric	☑
verification_status	string	categorical	☑
purpose	string	categorical	☑
dti	string	numeric	☑
delinq_2yrs	string	categorical	☑
inq_last_6mths	string	categorical	☑
open_acc	string	numeric	☑
pub_rec	string	categorical	☑
total_acc	string	categorical	☑

Your dataflow should look like this:



Compare Models

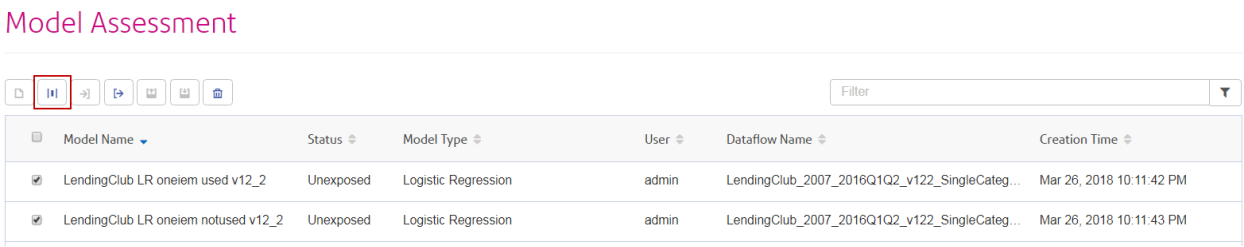
Because you just created two different Logistic Regression models, you can now compare the output of those models.

1. From the Model Output tab on either Logistic Regression stage, click the Output button, followed by the Model details link; it will lead you to the Machine Learning Model Management page.

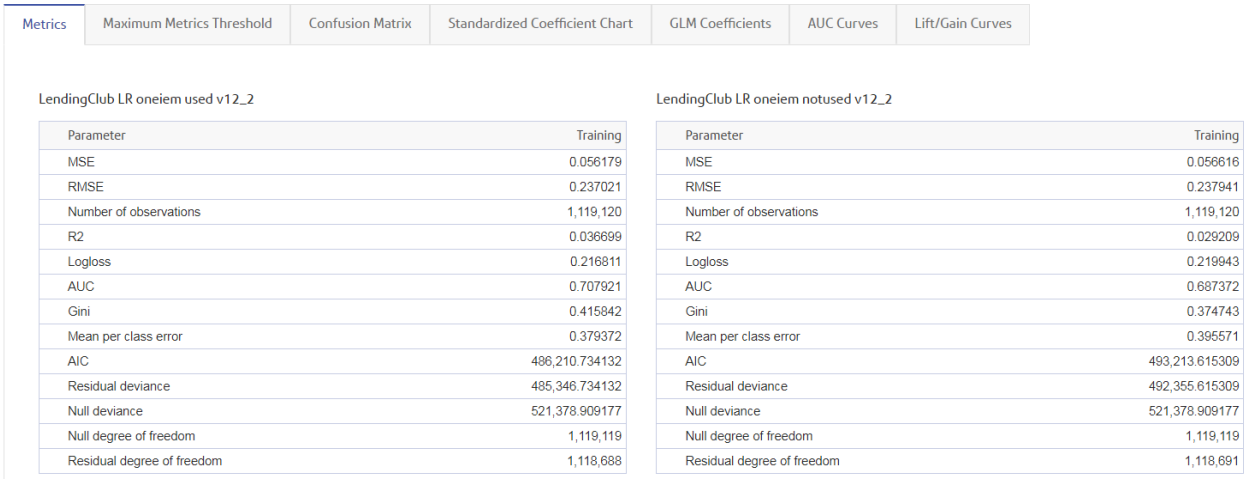
2. Click the **Model Assessment** tab.



3. Check the boxes next to LendingClub LR oneiem used v12_2 and LendingClub LR oneiem notused v12_2 (or 12_1, if appropriate), and then click the Compare Models ( ) button.



The comparison webpage will appear:



Random Forest Classification Modeling

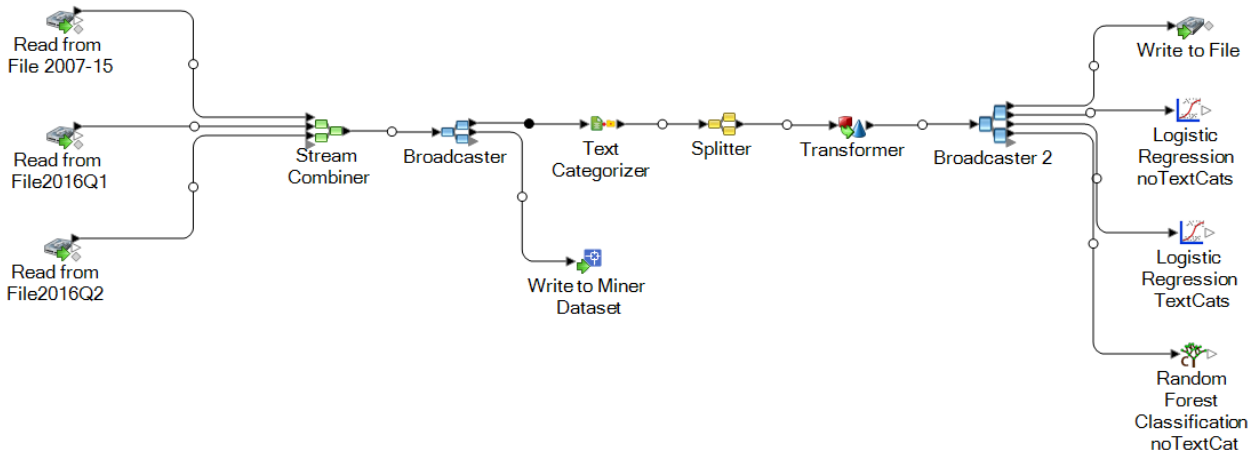
1. Drag a Random Forest Classification stage onto the canvas and connect it to the Broadcaster stage.
2. Double-click the Random Forest Classification stage; the Model Properties tab will appear.
3. Specify the model name as “LendingClub DRF oneiem notused v12_2” in the **Model name** field (or “LendingClub DRF oneiem notused v12_1” if you are running this demonstration on Spectrum™ Technology Platform 12.1).
4. Select Loan_Default in the **Objective field**. Check the **Overwrite** box to overwrite the existing model.

5. Select the following variables, ensuring the proper Model Data Type:

Input Fields	Type	Model Data Type	Includ
Loan_Default	**integer**	**categorical**	☑
time_history	integer	numeric	☑
RevolUtilizRate	double	numeric	☑
employer_listed	integer	categorical	☑
empty_desc	integer	categorical	☑
delinq_ever	integer	categorical	☑
emp_length_na	integer	categorical	☑
emp_length_new	integer	numeric	☑
loan_amnt	string	numeric	☑
home_ownership	string	categorical	☑
annual_inc	string	categorical	☑
verification_status	string	categorical	☑
purpose	string	categorical	☑
dti	string	numeric	☑
delinq_2yrs	string	categorical	☑
inq_last_6mths	string	categorical	☑
open_acc	string	categorical	☑
pub_rec	string	categorical	☑
total_acc	string	categorical	☑

6. Continue to the Basic Options tab.
7. Change **Number of trees** to 20. Leave all other selections as default value.
8. Leave the value 100 as the **Percentage for train data**.
9. Continue to the Advanced Options tab.
10. Leave Ignore constant fields checked to skip fields that have the same value for each record.
11. Leave **Seed for algorithm and N fold** unchecked.
12. Click OK and save the job.

    Your job should look like this:

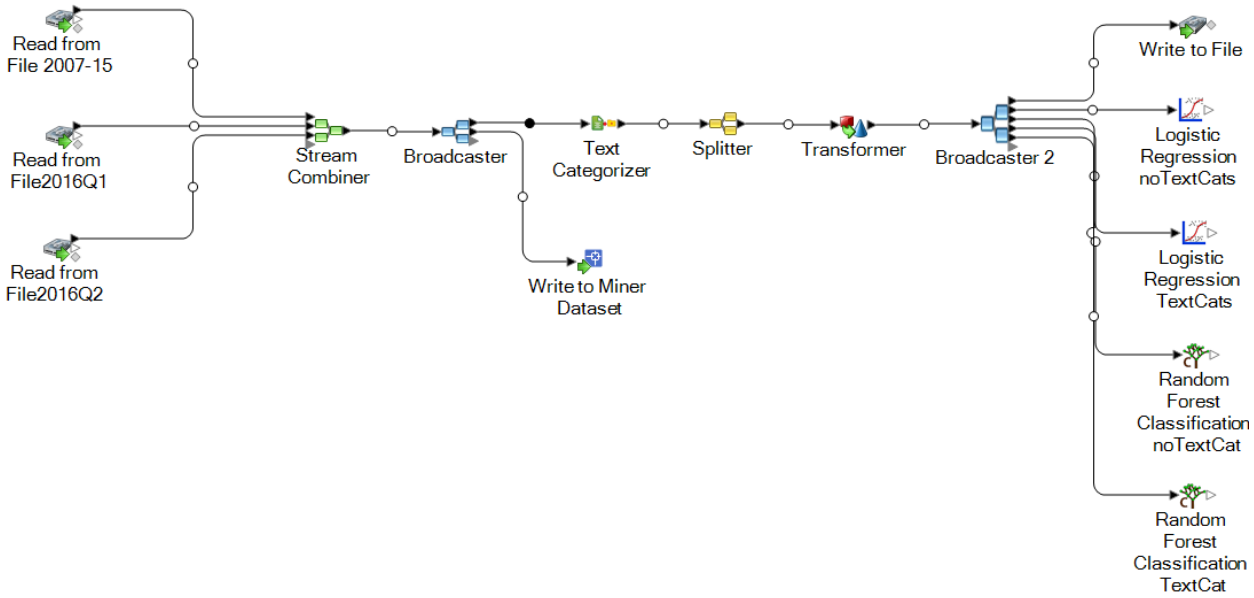


13. Click the Run button to run your dataflow. The Execution Details window appears and shows the status of the job.
14. Click Refresh. Once the status shows Succeeded, click Close.
15. Double-click the Random Forest Classification stage, then click the Model Output tab and the Output button. A limited version of the resulting model metrics will appear.

16. Click the Model details link; it will lead you to the Model Management Repository page.
17. If necessary, return to the model stage and change settings to train a better model.
18. Repeat steps 1-12 to create a second Random Forest Classification stage. This time we want to use an input predictor from the Text Categorizer stage.
    a. In the **Model name** field, specify model named "LendingClub DRF oneiem used v12_2" (or "LendingClub DRF oneiem used v12_1" if you are running this demonstration on Spectrum™ Technology Platform 12.1).
    b. Select the following variables, ensuring the proper Model Data Type. Note the addition of the descCategory field.

Input Fields	Type	Model Data Type	Include
Loan_Default	**integer**	**categorical**	☑
time_history	integer	numeric	☑
RevolUtilizRate	double	numeric	☑
employer_listed	integer	categorical	☑
empty_desc	integer	categorical	☑
delinq_ever	integer	categorical	☑
emp_length_na	integer	categorical	☑
emp_length_new	integer	numeric	☑
descCategory	string	categorical	☑
loan_amnt	string	numeric	☑
home_ownership	string	categorical	☑
annual_inc	string	categorical	☑
verification_status	string	categorical	☑
purpose	string	categorical	☑
dti	string	numeric	☑
delinq_2yrs	string	categorical	☑
inq_last_6mths	string	categorical	☑
open_acc	string	categorical	☑
pub_rec	string	categorical	☑
total_acc	string	categorical	☑

Your dataflow should look like this:



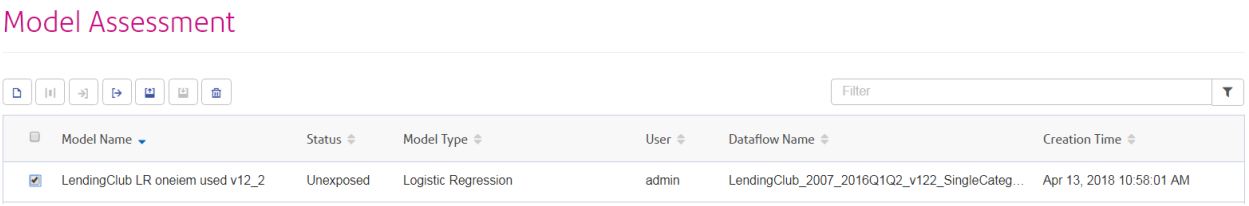
If you are satisfied with the new model, save the job with a new job name. As with the Logistic Regression stages, you can compare the output of the two Random Forest Classification stages.

Score New Data Using Java Model Soring

To score the new data we first need to expose the model.

1. Return to the Machine Learning Model Management page and click the Model Assessment tab. (Note that if you are using Spectrum™ Technology Platform 12.1, this is called the Model Analysis tab.)

2. Check the model you intend to expose.



3. Click the Expose button () and click Yes at the prompt. The model is now exposed.

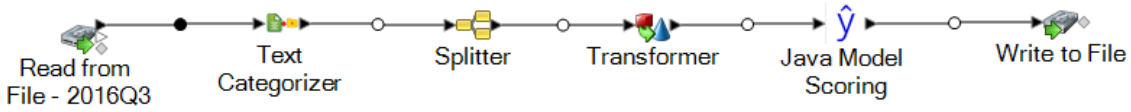
Second, we need to import the 2016Q3 data. Follow the same steps you took when importing the Q1 and Q2 data, being sure to remove any extraneous lines at the beginning or ending of the data file.

Third, we need to create a new dataflow that scores the model.

1. Open Enterprise Designer.
2. Select File > New > Dataflow > Job.
3. Drag a Read from File stage onto the canvas and double-click that stage.
    a. In the **File name** field, locate and specify the file *LoanStats_2016Q3.csv*.
    b. In the **Record type** field, choose Delimited.
    c. In the **Field separator** field, select Comma (,).
    d. In the **Text qualifier** field, select Double quote (“)
    e. In the **Record separator** field, select Unix (U+000A).
    f. Check **First row is header record**.
    g. Uncheck **Treat records with fewer fields than defined as malformed** and click OK.
4. Select Edit > Job options and check **Do not terminate the job on a malformed record** and click OK.
5. Select Edit > Type Conversion Options and check **Override system default options with the following values**. Select **Initialize the field using default values**, uncheck all **Fail Null** boxes, and then click OK.
6. Copy the Text Categorizer stage from the previous job and paste it into this job; connect it to the Read from File stage.

7. Copy the Splitter stage from the previous job and paste it into this job; connect it to the Text Categorizer stage.
8. Copy the Transformer stage from the previous job and paste it into this job; connect it to the Splitter stage.
9. Drag a Java Model Scoring stage onto the canvas and connect it to the Transformer stage. If you are using Spectrum™ Technology Platform 12.1, this stage can be found in the Machine Learning Module stages; if you are using 12.2 or later, this stage can be found in the Analytics Scoring Module stages.
10. Double-click the Java Model Scoring stage.
    a. Select the **Type filter** being used to score the model. For this demonstration we are using Logistic Regression.
    b. Select the **Model name** from the drop-down.
    c. Check **Ignore unknown categorical levels**.
    d. The **Inputs** table shows information for the model's input fields. These fields and their data types automatically map to Spectrum fields and data types.
    e. Click Model Output tab and click **Include** for each field whose data you want included in the model's output.
    f. Click OK to save the model.
11. Drag a Write to File stage onto the canvas and connect it to the Java Model Scoring stage.
12. Double-click the Write to File stage.
    a. In the **File name** field, navigate to the location where you want to store the file and call it “LoanStats_2016Q3_predictions_output.csv”.
    b. In the **Record type** field, choose Delimited.
    c. In the **Field separator** field, select Comma (,).
    d. In the **Text qualifier** field, select Double quote (“)
    e. In the **Record separator** field, select Unix (U+000A).
    f. Check **First row is header record**.
    g. Click the Fields tab, then Quick Add, then Select All, and then click OK twice.
    h. Click OK and save the job.

    Your dataflow should look like this:



13. View the output file to see the scored data.

Congratulations! You have finished the demonstration!