Practice: Using the Python API to Explore Movie Description Text Documents

The data set **MOVIE_MONEY** contains a description of popular movies. Each overview is paired with a profit indicator indicating if the movie made more money than its budget. The goal of this analysis is first to analyze the raw unstructured text and then build a model to predict if the movie is profitable based solely on its description.

Name	Model Role	Measurement Level	Description
PROFIT	Target	Binary	1 = movie made more money than the budget, 0 = otherwise
TITLE	Nominal	Text	Movie Title
OVERVIEW	Input	Text	Description of the movie

1. From the Jupyter Lab Home directory, select the plus symbol to open a new page and then select Python under Notebook to create a new notebook.

2. Load the os, sys, SWAT, numpy, pandas, and matplotlib packages. Use the %matplotlib inline statement to create graphics within the notebook and set the CAS option to print CAS messages in the notebook.

3. Connect to CAS and create a connection object called **conn**.

4. Use the upload action to load the **movie_money.sas7bdat** file onto the CAS server and name the CAS table **movies**. Create a table reference using defCasTable and then print the dimension and head using SWAT functionality.

5. Use the runCode action from the dataStep action set to give the data a minor clean. Keep only letters and reduce them to lowercase. Also add a sequence to the CAS table to create unique document identifications for each observation. Print the first few observations to view the changes.

6. Use a DATA step to find all the documents that include the term *denzel* and save them to a new CAS table. Print the first few of these observations and the total number of documents found. Next, do the same except find the term *karate*.

7. Use the tpParse action from the textParse action set to parse the terms in the documents. Use stemming, and avoid noun groups, entities, and tagging. Save the parse configuration and offset information as new data sets.

8. Upload the list of stop words in the **stop_words.csv** file from the demonstration and print the total number of stop words.

9. Use the tpAccumulate action from the textParse action set to accumulate terms using the stop list. Use stemming, avoid tagging, and set reduce to a value of 1 to keep all terms in the collection. Save the parent, child, and terms tables as well.

10. Create a reference to the **terms** data table and then use SWAT functionality to find the total number of terms and total number of unique terms.

11. Use a DATA step to create a new CAS table with only the unique terms.

12. Use the execDirect action from the fedSql action set to create a table with the top 100 most used terms.

13. Download the top 100 terms to the client and then create a bar chart of the top five most used terms.

14. Use the tMine action from the textMining action set to find 10 topics in the movie collection.
    - use the uploaded stopList and stemming
    - avoid noun groups and tagging
    - set reduce to a value of 1 to keep all unique terms in the analysis
    - find ten topics and ten labels for each and categorize each movie into one of the topics using the topicDecision argument
    - save the docpro and topics tables
    - print the ten extracted topics.

    Note: The second half of this notebook builds deep learning models to predict when consumers dispute the company response based on the complaint. If you end this CAS session, you'll need to perform a few extra steps before you can begin the lesson 4 practice.