**Auto classify Documents in SharePoint using Azure Machine learning Studio: Part 2**

[**AZURE**](http://www.sharepointtweaks.com/search/label/Azure?max-results=6)  TUESDAY, APRIL 17, 2018

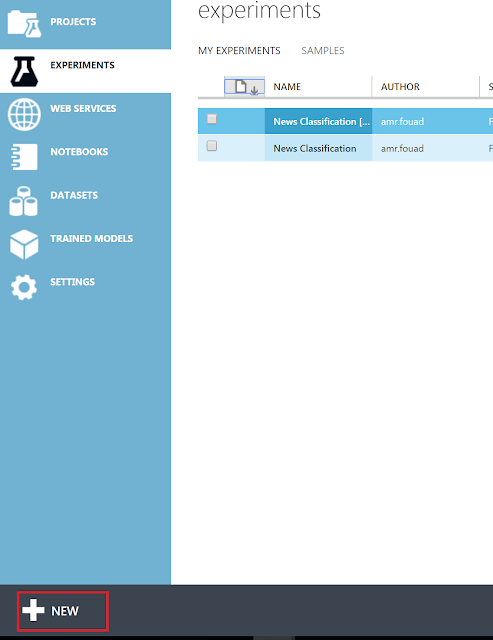
This is part 2 of a two parts blog series which explains briefly how to use azure machine learning to auto classify SharePoint documents. In part [one](http://www.sharepointtweaks.com/2018/04/auto-classify-Office365-content-using-azure-machine-learning-studio-part1.html), we covered the end to end solution skeleton, which relies on using Microsoft flow. The flow is set to be triggered whenever a new document is uploaded to our target SharePoint library.  
  
The main challenge we faced is how to extract text representation from the Microsoft office file .docx and as explained in the previous blog post. I end up using the .NET version of Open Source [Tika](https://kevm.github.io/tikaondotnet/)to extract the text. In the previous blog post we referred to the Azure Machine learning as web service we call using flow HTTP invoke action. Today we will explore this black box in more details:

1. **The Data**

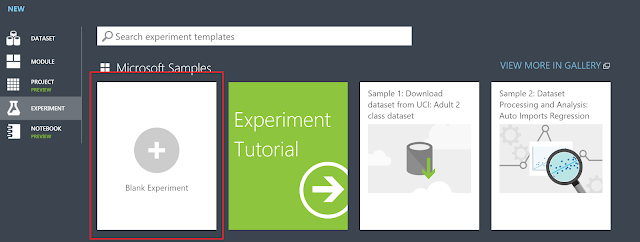
This is by far the most important and complicated step of the whole process as it's too specific to the problem you're trying to solve. Also the availability of data that is sufficient to train your model with high probability of correctness is something you will spend most of the time trying to figure out.  
In my example as it's merely a POC, I did choose an easy path. I used an existing dataset available to everyone has access to Azure ML studio (BBCNews Dataset) and then I tailored the SharePoint content to match the data set.

1. **The Training Experiment**

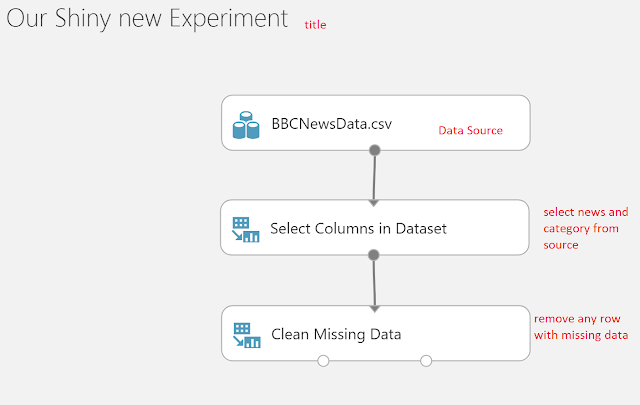
Create your first Azure ML training experiment is a relatively easy task compared to data preparation phase.   
First step is to navigate to <https://studio.azureml.net/> and login using your Microsoft Account, you don't need to have an active Microsoft Azure subscription or credit card. LUIS and Bot framework were also free but now you need to have Azure subscription to use these two other services, don't know whether this will change in the future. However, till the time of writing this blog post it's still free.  
From the left menu choose experiments and choose new , choose blank experiment as we will build this together from scratch

[](https://2.bp.blogspot.com/-qubQpnncY4M/WtWaG_SFKaI/AAAAAAAAC5c/LqpLXiJv0KoH_jatoC56MqyRt5eGv4MJACLcBGAs/s1600/1.png)

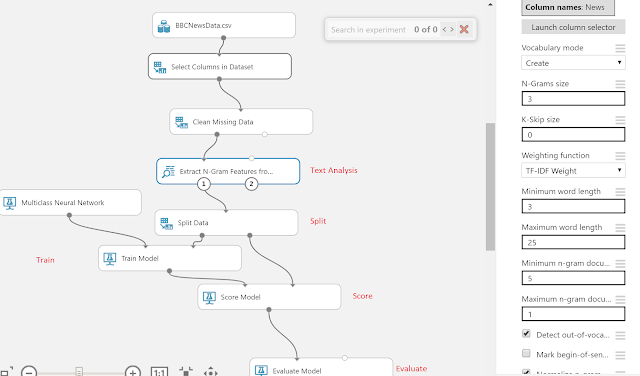
Let's name our experiment (our shiny new experiment) and we will use BBC news data set, you can substitute this with your own prepared set which will have the text representation of Office document with the current category

[](https://3.bp.blogspot.com/-sT_mvrHJFYc/WtWaHVGCWAI/AAAAAAAAC5k/HZsvRY2YtZU57IisopzzPOweER2eGqqvQCEwYBhgL/s1600/2.png)

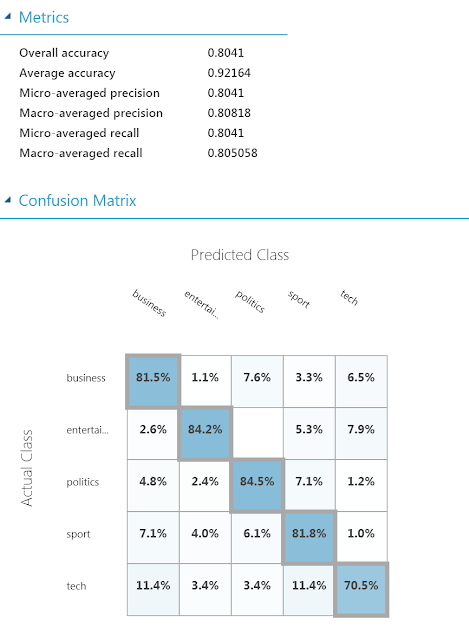
Then we will do some data cleansing, select News and category from the data set and clean up empty rows by removing any rows that does have any missing columns by setting minimum missing values range 0 to 1 ( that means if only single column has a missing value the action "cleaning mode" will be triggered , we will choose to remove the entire row

[](https://2.bp.blogspot.com/-5sDzpjo1v1E/WtWaIQMSJzI/AAAAAAAAC5o/ZrLw2XMAh_g0u6F0-q7SdZMWukqMKgOzQCEwYBhgL/s1600/3.png)

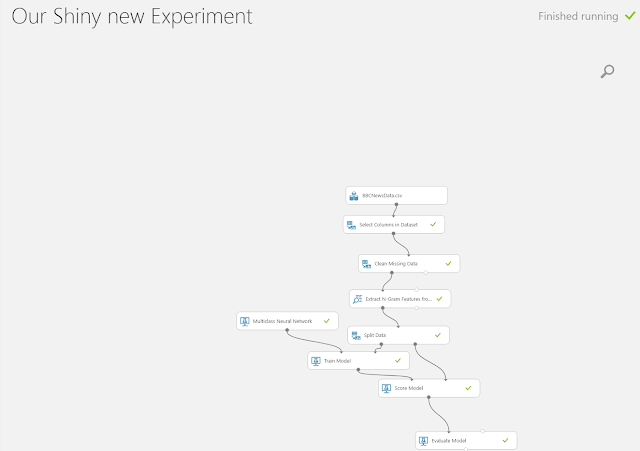
Now it's getting more exciting, we will use some text analysis technique called Extract N-Gram features, this will use the News column (text representation of the document) as an input and based on the repetition of a single word or tuple of words it can analyse the tuple effect in the categorization.  
First thing we will change the default column selection from any string column to only analyse the News column which represents the text extraction from Office document.  
  
Secondly,  we will choose to create a vocabulary mode, the result vocabulary can be used later on in the predictive experiment. the next option is the N-Gram size which dictate to what extend you want the tuple to grow. for example if you keep the default 1 it will only consider a single word. However, if you choose 2 it will consider single word and any couple of consecutive words. In our example we will use three that means the text analysis will consider a single word, two words and three words tuples.   
In the weighting function there are multiple options, I will choose TF-IDF which means term frequency and inverse document frequency.   
  
This technique is give more weight for terms that appears more than others with the consideration of giving a negative score for terms that appears in different documents( news items in our case) with different classification. The final score for the vocabulary is a mixture of both TF and IDF values.  
  
There are lots of other options which can be viewed in more details in this excellent guide for N-Gram module guide [here](https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/extract-n-gram-features-from-text)  
  
One important option is the desired output features which is basically how many tuples you want to use to categorize your data, this might be a trial and error for a newbie like me until I can see the top n effective tuples to make it easier and faster for the trained model to compare future text against, in my scenario I used 5000 Features as a result of N-Gram Feature extraction step.  
  
After the data preparation and vocabulary preparation , we will use 4 steps that is common across any training experiment which are (Split, Train, Score and Evaluate)  
  
The first Step is to split the data randomly with some condition or purely based on a random seed, train the model using part of the data then score the model using the other portion, last step is to evaluate the model and view  how accurate your model is.

[](https://1.bp.blogspot.com/-4ROgn1z1trM/WtWaIabBQHI/AAAAAAAAC5s/Peh43HvvKYwWseo48It9m8XJ8zjLUTqWACEwYBhgL/s1600/4.png)

Based on the evaluation result you can see overall model accuracy, at this point if the model is not hitting the mark ( you can set target accuracy based on business requirements). One possible solution is to change N-Gram tuple values, output features or even use a complete different training algorithms. Sometimes it's useful to to include additional columns or metadata to help categorize the documents maybe a department name or author not only the document content.   
In my sample I got overall accuracy of 80% which is accepted to me so relying on the text extraction from the documents is sufficient to me.

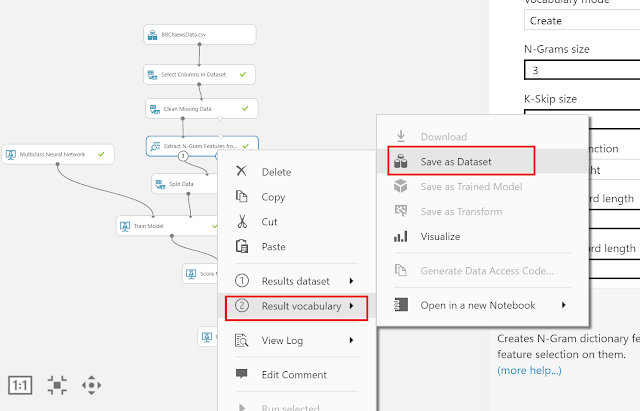
[](https://1.bp.blogspot.com/-BqO_9ohyj5c/WtWaIoy_3MI/AAAAAAAAC5w/Jk75r881KFYt8fldWM7AWubYrLKAkAxgQCEwYBhgL/s1600/5.PNG)

Now let's run the experiment and confirm that all steps are executed successfully which we can validate with the green check mark on every step

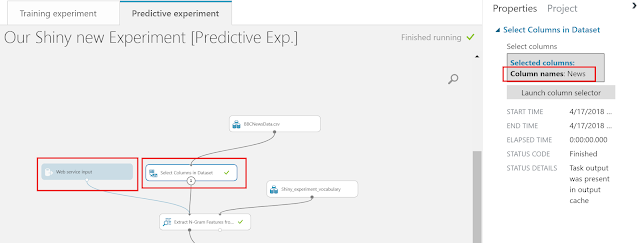
[](https://1.bp.blogspot.com/-AGXLjF_Cks8/WtWaJdSBkMI/AAAAAAAAC50/30lq_a69f_EXaETnR8gjrbrk4vDDli2WgCEwYBhgL/s1600/6.PNG)

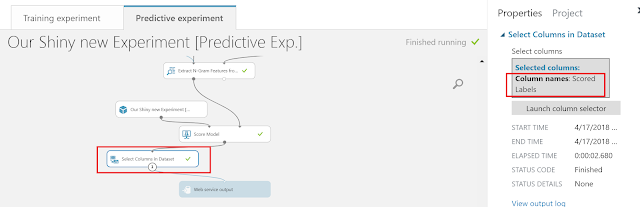
1. **The Predictive Experiment**

After a successful run of the training experiment we can easily hover over setup a web-service in this case we can have retaining web service which will allow use to provide Dataset as an input and spit out the trained model and evaluation results as an output. for the sake of this blog post we will not play around with this option. We will instead generate a predictive experiment which will allow us to predict item classification based on its text representation. So using the same button in the lower toolbar we will choose to create predictive web service this will generate the predictive experiment for us if we run this and publish it, we will cause plenty of errors as the generated steps trying to create new vocabulary too.  
Let's use the generated vocabulary of the training experiment as an input (we need to save the result vocabulary of the training experiment as Data-set so we ca use it later)

[](https://2.bp.blogspot.com/-ffLR6DnTRDw/WtWaJkHbGuI/AAAAAAAAC6Y/H6y44Laf2L0blaWtRyfAqhB6JebCXqjBgCEwYBhgL/s1600/7.png)

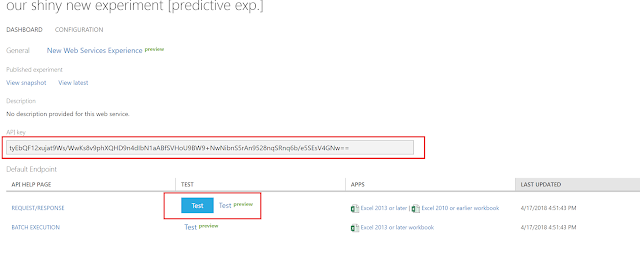
We will also remove any transformation steps , we will only select single column (news text representation) which will be the single input for our web service.

[](https://4.bp.blogspot.com/-__xIjko_EVU/WtWaJ9Vc0uI/AAAAAAAAC6M/0hoEUFs5uR8yIQ_9zUeo-jDNE3vXFB7dQCEwYBhgL/s1600/8.png)

[For the webservice output we will also select single output which will be scored label   
[](https://2.bp.blogspot.com/-cDtHiQihFhs/WtWaKcOj8BI/AAAAAAAAC6Y/3a9s_Lnk2xEcvwbWIum9Zy7u42tG3zjcgCEwYBhgL/s1600/9.png)](https://2.bp.blogspot.com/-cDtHiQihFhs/WtWaKcOj8BI/AAAAAAAAC6Y/3a9s_Lnk2xEcvwbWIum9Zy7u42tG3zjcgCEwYBhgL/s1600/9.png)

1. **The Webservice**

Let's run the predictive experiment now and make sure it's working, then we can deploy the web-service which we can use to classify text representation of documents. This will open a new page which allows us to test our web service by supplying text as "News" input and provide scored Label output as text.  
  
P.S. you must pass the API key as authentication header to make it work ;)

[](https://1.bp.blogspot.com/-4D2uC6F-c04/WtWaHNjI29I/AAAAAAAAC6Q/K8l4Uaj0IJQdAljpdJ9B1ahFh1c14O4JACEwYBhgL/s1600/10.png)