**Sentiment Analysis On PokemonGo**

**Objective:**

To use algorithm for Prediction, Classification and Clustering of Tweet Sentiments using Microsoft Azure and deploy the project into a web service using AWS

**Data Cleansing:**

Following are the steps we made to cleanse the pokemon dataset using RStudio.

1. Remove punctuations, control characters and digits.
2. Split sentence into words with str\_split function from stringr package
3. Compare words to the dictionaries of positive & negative terms
4. Clean up sentences with R's regex-driven global substitute, gsub() and convert all the sentence to lower case
5. Remove retweet entities, @people, numbers, unnecessary spaces and html links
6. Classify your emotion and get the best fit
7. Substitute NA's by "unknown" and classify your polarity
8. Show your data frame with results
9. Plot distribution of emotions

**Running the model:**

1. **Binary Classification Model**

**Step 1:** Get data

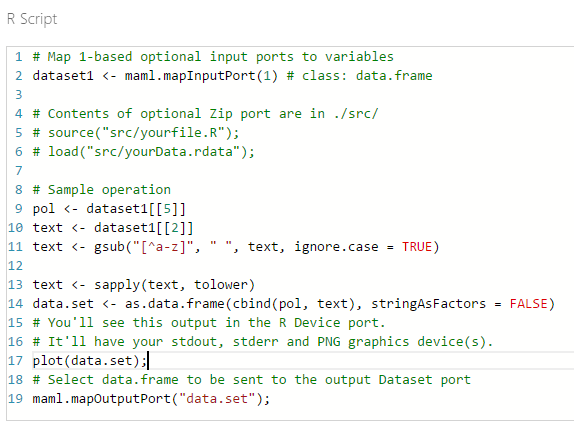
Upload the CSV file of your pokemon dataset. The CSV file should be cleansed with the steps mentioned above.

Each instance in the data set has the following fields:

* pol - the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
* tweet\_id - the id of the tweet
* time\_stamp - the date of the tweet (Sat May 16 23:58:44 UTC 2009)
* user\_id - the user who posted the tweet
* tweet\_text - the text of the tweet
* Sentiment\_score - the sentiment score of each tweet based on its polarity

**Step 2:** Text preprocessing using R

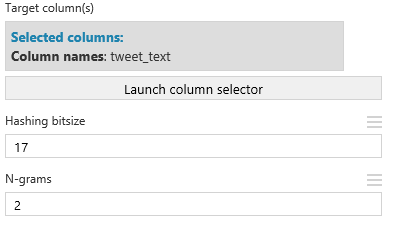
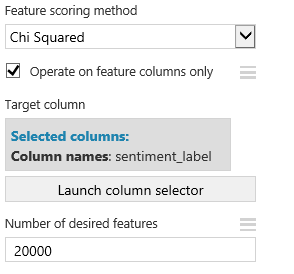
Unstructured text such as a tweets usually requires some preprocessing before it can be analyzed. We used the following R code to remove punctuation marks, special character and digits, and then performed case normalization:



After the text was cleaned, we used the **Metadata Editor** module to change the metadata of the text column as follows.

* We marked the text column as non-categorical column.
* We also marked the text column as a non-feature.

**Step 3:** Feature engineering

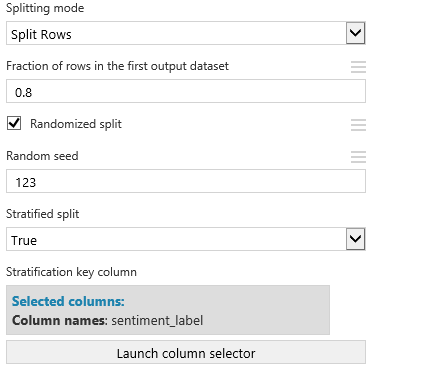
 

we used the **Filter Based Feature Selection** module to select a compact feature subset from the exhaustive list of extracted hashing features. The aim is to reduce the computational complexity without affecting classification accuracy.

We chose the Chi-squared score function to rank the hashing features in descending order, and returned the top 20,000 most relevant features with respect to the sentiment label, out of the 2^17 extracted features.

**Step 4:** Split the data into train and test

The Split module in Azure ML is used to split the data into train and test sets where the split is stratified. The stratification will maintain the class ratios into the two output groups. We use the first 80% of the pokemon sample tweets for training and the remaining 20% for testing the performance of the trained model.

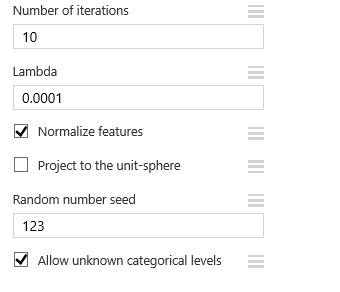


In the launch column selector, you will be using the variable called pol.

**Step 5:** Train prediction model

To train the model, we connected the text features created in the previous steps (the training data) to the**\*Train Model** module. Microsoft Azure Machine Learning Studio supports a number of learning algorithms but we select SVM for illustration.

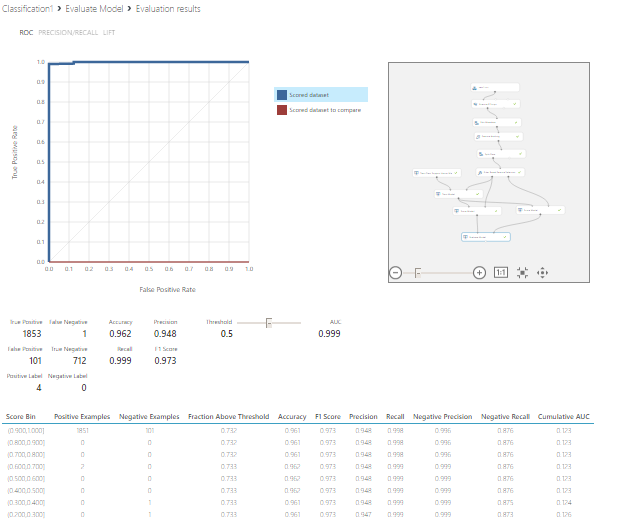
Two class support Vector entries:



**Step 6:** Evaluate model performance

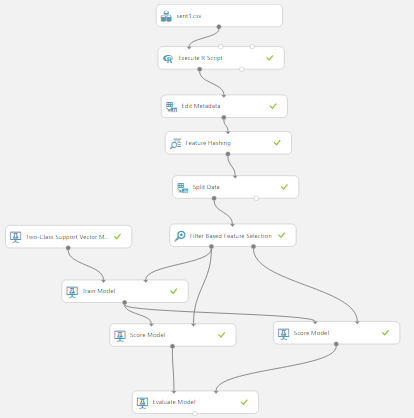
In order to evaluate the generalization ability of the trained Support Vector Machine model on unseen data, the output model and the test data set are connected to the Score Model module in order to score the tweets of the test set. Then connect the out predictions to the Evaluate Model module in order to get a number of performance evaluation metrics as shown below.

Finally, we added the **Evaluate Model** module, to get the evaluation metrics (ROC, precision/recall, and lift) shown in the following charts.



**Step 7:** Publish prediction web service

A key feature of Azure Machine Learning is the ability to easily publish models as web services on Windows Azure. In order to publish the trained sentiment prediction model, first we must save the trained model. To do this, just click the output port of the **Train Model** module and select **Save as Trained Model**.



Classification Model

1. **Clustering**

**Step 1:** Get data

Same as steps repeated for classification model

**Step 2:** Text preprocessing using R

Same as steps repeated for classification model

**Step3:** K-means Clustering

This experiment demonstrates how to use the K-Means clustering algorithm to perform segmentation on sentiment’s polarity from the pokemon dataset, based on the text of emotions about each tweet.

**Step 4:** Train prediction model

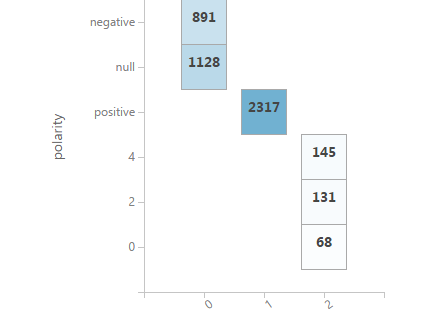
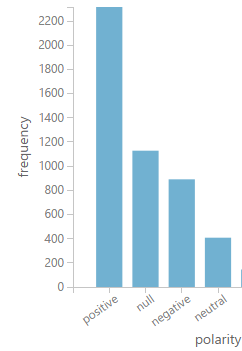
To train the model, we connected the text features created in the previous steps (the training data) to the**\*Train Model** module. Microsoft Azure Machine Learning Studio supports a number of learning algorithms but we select SVM for illustration.

**Step 5** Evaluate model performance

Once the data was prepared, we created several different instances of the **K-Means Clustering** module and trained models on the text data. By trial and error, we found that the best results were obtained with 2 clusters, but models using 1 and 3 clusters were also tried.

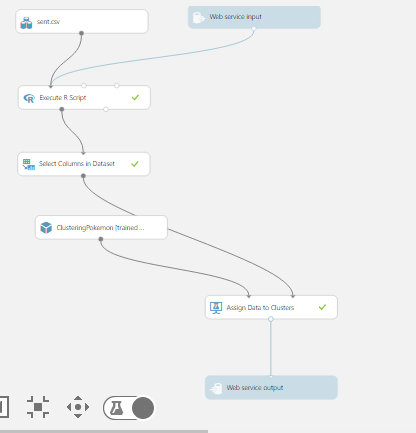
Finally, we used **Metadata Editor** to change the cluster labels into categorical values, and saved the results in CSV format for downloading, using **Convert to CSV** module.



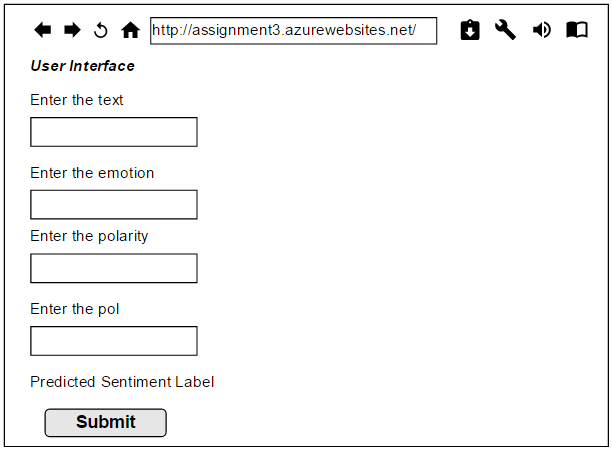
**Step 6:** Publish prediction web service

A key feature of Azure Machine Learning is the ability to easily publish models as web services on Windows Azure. In order to publish the trained sentiment prediction model, first we must save the trained model. To do this, just click the output port of the **Train Model** module and select **Save as Trained Model**.



**MOCKUPS: (app.moqups.com)**

***User Interface Page:***



***Workflow of Azure web services***

