Sentiment Analysis

Detecting Sentiment is one portion of overall [text analytics](https://github.com/Azure/azure-sdk-for-net/tree/master/sdk/textanalytics/Azure.AI.TextAnalytics).

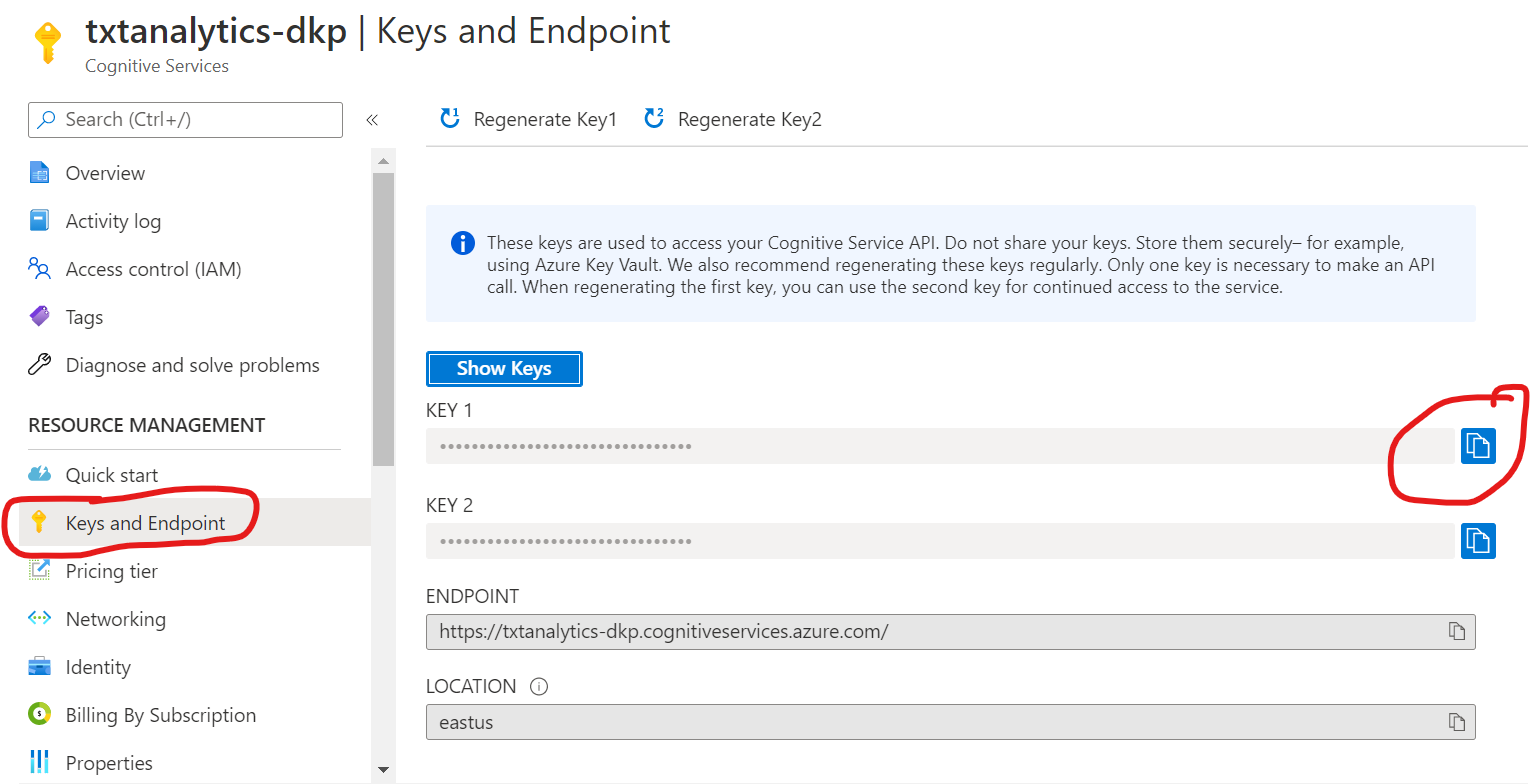
# Part-1

## Goal

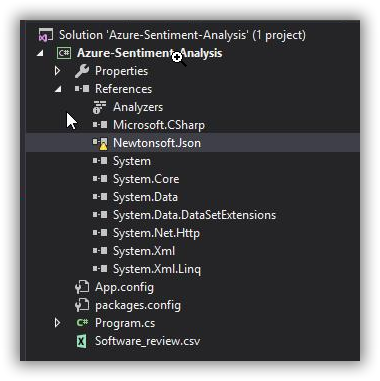
Find the sentiment of existing feedbacks quickly and classify as good and bad comments.

## Steps

1. For a quick solution, we are going to use Azure Cognitive Service.
2. Create a “Text Analytics” service in Azure Portal with Pricing Tier – “Free F0 (5K Transactions per 30 days)”. Go to that Resource’s page once created.
3. Download this sample C# console app -
4. Replace the access key and end point in **Program.cs** with the one you created.



1. Build the app to get the packages downloaded.



1. Run the app. You will find the sentiment of the feedback present in “**list\_reviewText**” column of table – “**Software\_5\_1\_100.sql**”
2. Imagine that, like this we have analysed the sentiment score of all the comments we have received. And it is saved as additional column “**sentiment**” in each row.

So now we have a dataset of all the feedbacks along with the sentiment analysed for the feedbacks. We are going to use this for train our new(-born) model.

# Part-2

## Goal

With the analysed dataset, deploy a quick dashboard that exposes the findings to take decisions by stake holders.

## Steps

1. **CONNECT –> 2. VISUALIZE –> 3. SHARE**

This format of working with a business intelligence tool will be almost common for any BI tool - *Power BI, Tableau, Qlik, Sisense*.

### Connect

1. I hope all of you would have created installed Power BI app in your machine.
2. Login with your personal Hotmail ID or azure-conf email assigned to you.
3. **Connect** to Microsoft SQL Server as **Live** mode.
4. Credentials of SQL Server:

|  |  |
| --- | --- |
| **Server name** | azconfdkp.database.windows.net |
| **Username** | dineshp |
| **Password** | Coolcomp@123 |
| **Mode** | Extract |
| **Database** | azconfdkp |
| **Table** | software\_feedbacks |

1. You have completed the first part - **enabling connection to data source**.

### Visualize

1. Next to visualize the data, create a new dashboard.
2. You will be having a blank dashboard. Now let us populate some widgets and configure data to the same.
3. Add a filter widget ‘Combo Box’ and bind ‘Rating’ column into it.
4. Similarly add another ‘Combo Box’ and bind ‘Sentiment’ column into it.
5. Create a column chart to list down *number of feedbacks ordered by rating.* Move the column ‘’ into Values and column ‘’ into Measures.
6. Add a Grid widget and populate Customer name, Order ID, Comments, Rating and Sentiment.   
   You have completed the second part – **Visualizing the data**.

### Share

1. Save the dashboard by publishing it.
2. Now the dashboard is available, and you can apply filters in Rating, Sentiment combo boxes and workout all the features like sharing dashboards, exporting the charts, etc.

With that completed the 2nd part – **Sharing the dashboard and insights**.

# Part-3

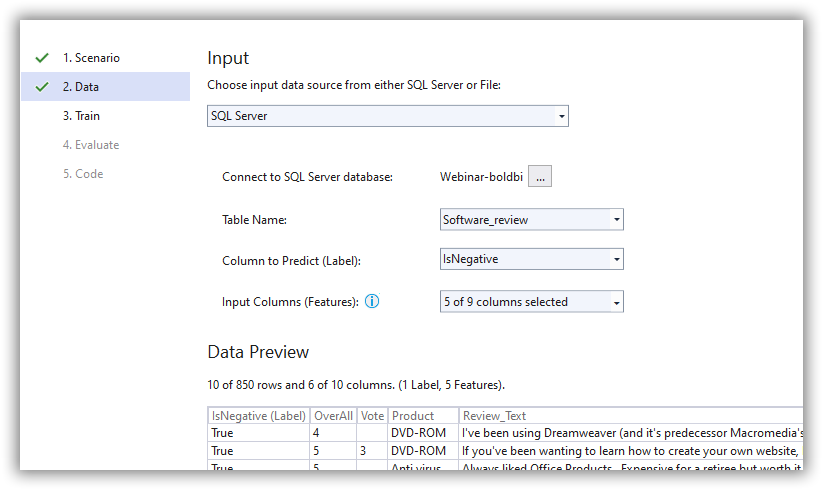
## Goal

Stack holders are satisfied with the sentiment analysis result and dashboard. They suggest to create a ML model with the data we have to find out good and bad feedbacks. *Engineering team* then deploy the process as a pipeline to find the sentiment score of new data, re-train the model to some point. At a stage, finding sentiment score with a cloud service will be stopped in total and the trained model will be used to find the sentiment score of future feedbacks directly.

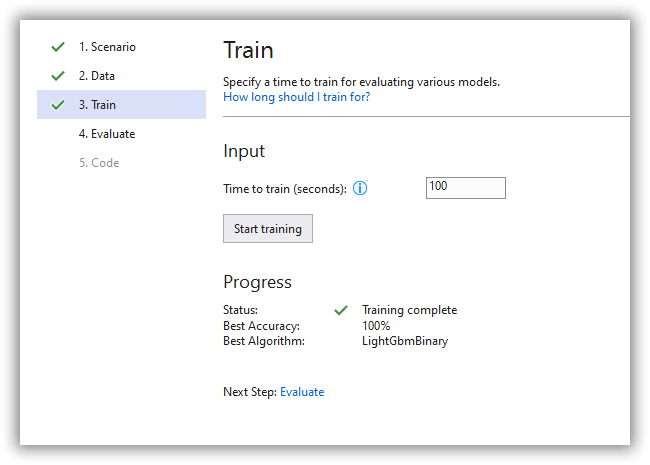
### Steps

Having the data and its defined sentiment result, dev team decides to deploy a new machine learning model. May not due to knowledge gap, but to implement it quicker, they choose ML.Net;

* If you are having Visual Studio 2017 15.9.12 or later, follow further to deploy your own new Machine Learning model for sentiment detection which can be used instead of Cognitive Service.
* Follow the instructions and install ML.Net Model Builder extension  
  <https://dotnet.microsoft.com/learn/ml-dotnet/get-started-tutorial/intro>
* In “Download and add data” – instead of file, you can choose the SQL Server and connect to the same ‘Software\_review’ table which you connected while creating dashboard.
* Under **Column to predict (Label)**, select "sentiment".
* **Input Columns** – Uncheck columns ‘list\_reviewerID, ‘list\_verified’, ‘list\_asin’, ‘list\_reviewerName’.



* Change the Time as 50 seconds based on need. Here the default is fine.



* Proceed Evaluate and Code.
* Run the console app and see that it has evaluated the first row from data with the new Binary-Classification model created.
* Now you can publish the Model and deploy it in a web app or Azure Function and consume it anywhere. Because this deployment is in .Net Standard

# Learnings:

* Model Builder’s model creation **doesn’t require internet**. Anyway while using Model builder, the code is generated without assemblies. For assemblies need internet.
* Best fit for .Net based projects with machine learning use case.
* Prepared model in .Net core language can also be published to use in all 3 major OS – Linux, Windows, Mac. Code once – Deploy anywhere.
* Model Builder is evolving so fast to do other operations of building a model via ML.Net like,
  + Data Preparation,
  + Evaluate intermediate data in the output of each stage in the machine learning pipeline.
* Model Builder limitations:
  + SQL Server to have 1 lakh rows as max to train,
  + Train in Local most of the cases.
* ML.Net limitations: (Include for Model Builder as well)
  + Retraining a model is possible (for specific algorithms as of now).
  + If there is New line (\n) in data ML.net will fail to train and show error 'Unable to split file provided into multiple consistent columns'. Solution - As of now, replace new line with a character like space.