CS4830 Final Project (Yelp Dataset)

Name	Roll Number
Ayush Toshniwal	EE17B157
Pranav Pendharkar	CH17B115
Aryan Pandey	CH17B105

Objective

To use Natural Language Processing to predict the ratings posted by various users on the Yelp website in terms of stars. The above problem statement has to be solved using:

- Batch Computation: Pre-processing on the dataset and training models in DataProc cluster using Spark
- Real-time Computation: Using best trained model to perform real-time predictions of test data streaming to Kafka

Pre-processing

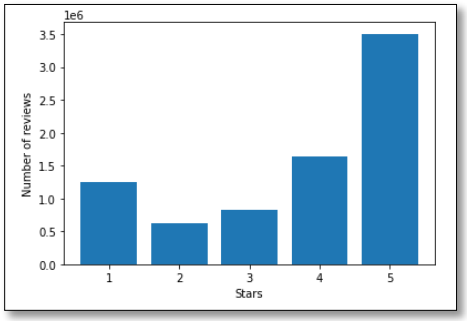
The given training dataset – "**gs://bdl2021_final_project/yelp_train.json**" contains 48 sub-files and a total of 7,863,924 (more than 7 million) reviews by various users.

The dataset has a total of 9 columns, including the target column: **"stars"**

	business_id	cool	date	funny	review_id	stars	text	useful	user_id
0	-0KMvRFwDWdVBeTpT11iHw	0	2016-12-29 20:01:49	0	jPrUWCv6TzHHVqIlBwHwAg	2	Great food ok price. The problem with this pla...	0	XWjRrDyLm2QFV7M8URP9og
1	-0aOudcaAyac0VJbMX-L1g	0	2014-10-20 16:06:09	0	BNyI8G0PCBLQIjAeXI0qMw	3	Local hole in the wall pizza and shawarma plac...	4	4KbUaerUBoiUwZGMHKVFxQ
2	-0aOudcaAyac0VJbMX-L1g	0	2015-03-23 00:39:25	0	Sosaq-6xKHCmgTzscSTbGw	4	I'm giving this place a four stars for the qua...	4	Zy7NmPA8HeiBccqezhqJrA
3	-0aOudcaAyac0VJbMX-L1g	0	2016-03-22 03:43:36	0	muJ8DynRRZCwFaEMgnk3nQ	5	Really good pizza and fries! & panini! Super b...	0	R0jQ-a5V1GzgfELO7jna4g
4	-0mgn9dRi95otXd_h61LdA	0	2016-08-20 04:54:24	0	EsUTAL_KRFXwDg4ezt0zQQ	1	One of the worst shisha places in toronto. I t...	0	wd0uUNvtEFDfNzol44Om-w

Data Exploration

- **'Stars'**



Stars	Number of reviews
1.0	1,253,755
2.0	621,421
3.0	824,152
4.0	1,637,783
5.0	3,506,247

Average length of a review	84.428 words
Number of categories in 'cool'	198
Number of categories in 'funny'	206
Number of categories in 'useful'	263

Combining JSON Files

The entire dataset (approx. 6GB) is broken down into 48 files. So, the first step is to combine all JSON files to a master dataframe. To facilitate the concatenation, **map-reduce** technique is used.

Preprocessing on text column

- Tokenization: Separating a piece of text into corresponding words, which can be easier to process in next steps. **RegexTokenizer** is used to tokenize the text into words.
- Stop words removal: Filtering of stop words (words that do not add much meaning to a sentence) is done using **StopWordsRemover** from Pyspark ML-Feature library.
- Count Vectorizer: Converting the above collection of text documents into a sparse representation.

```
+--------------------+--------------------+--------------------+
|               words|            filtered|                  cV|
+--------------------+--------------------+--------------------+
|[i, m, not, typic...|[m, typically, fa...|(10000,[5,6,15,19...|
|[i, saw, hamid, a...|[saw, hamid, tony...|(10000,[9,10,17,2...|
|[kayla, is, the, ...|[kayla, girl, wan...|(10000,[16,31,42,...|
|[fast, and, frien...|[fast, friendly, ...|(10000,[4,7,15,30...|
|[in, their, infin...|[infinite, wisdom...|(10000,[2,4,6,34,...|
+--------------------+--------------------+--------------------+
```

"words": Represents the tokens after tokenization on "text' column

"filtered" column does not have stop words

"cV" represents the output column of the CountVectorizer

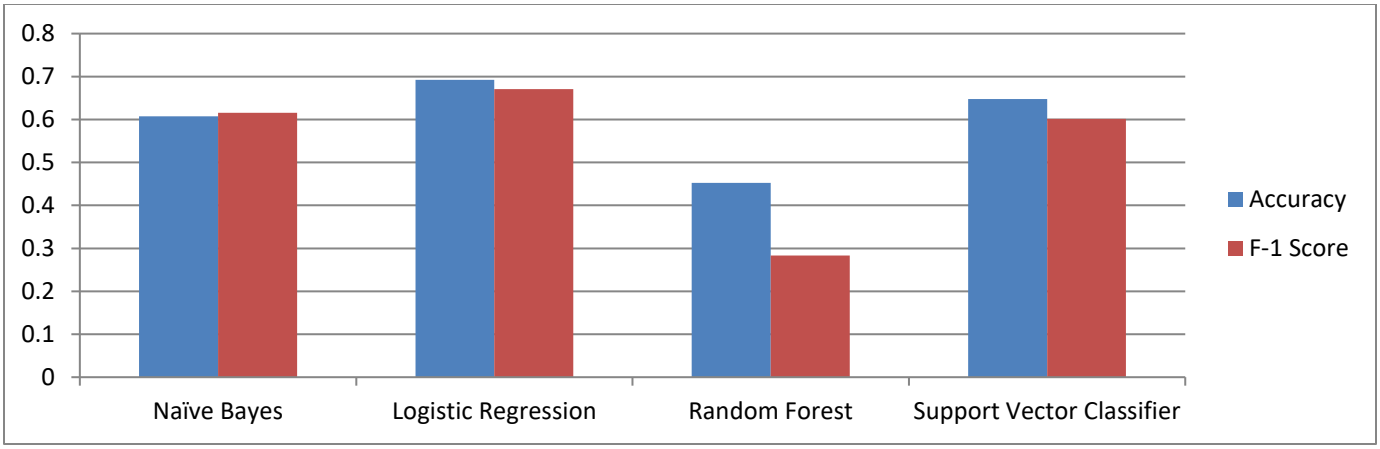
Model Fitting

For identifying best model, we tried out 4 different algorithms (Logistic Regression, Naïve Bayes, SVC and Random Forest) along with addition/elimination of features ("funny", "useful")

Below table summarizes our findings:

Model	Training Dataset		Test Dataset	
	Accuracy	F-1	Accuracy	F-1
Naïve Bayes (Only text)	0.6089	0.6167	0.6075	0.6155
Naïve Bayes (Text + funny + useful)	0.6088	0.6168	0.6077	0.6158
Logistic Regression (Text + funny + useful)	**0.6984**	**0.6775**	**0.6923**	**0.6709**
Random Forest (Text + funny + useful)	0.4515	0.2824	0.4521	0.2831
Support Vector Classifier (Text + funny + useful)	0.6621	0.6192	0.6478	0.6014

Below is the graph for Accuracy and F1-Score for Testing Dataset



So, the best model according to our fitting was Logistic Regression. While we wanted to further tune the parameters, and find the most optimal model, it took enormous amount of time to fit because of the size of dataset (> 6GB)

It was worthwhile to note that Random Forest underperformed despite its reputation as a robust and versatile method. We reason out this because RF is not the best choice for high-dimensional sparse data.

Real-Time Computation

- Producer Screenshot for one data point


```
b'{"business_id":"0WTAkJNPznTgmiOUTue9NA","cool":1,"date":"2008-09-15T16:51:21.000Z","funny":0,"review_id":"67zD0RNAo38T-9HEw3fqJA","stars":4.0,"text":"Convention Centers can
strange beasts full of dragons and wicked men peddling drinks and food for far too much money. The Palais des Congres is not one of those, but a clean and wonderfully staffed
location where I\'ve twice been for conferences (and will be returning to for WorldCon next year)\\n\\nThe Tim Horton\'s is the biggest plus. I love it and the price is right.
 The Convention Center mark-up seems to be much smaller than for places like Starbucks when they\'re put into CCs. \\n\\nThe place is comfortable, and when I lost a bag, I sim
ply asked one of the roamers and they sent folks out to find it and finally they did. After less than 10 minutes. And they brought it to me in a panel I was attending! Good pe
ople.","useful":2,"user_id":"MIKByQgLWgXNU_26CX1Acw"}'
```


- Subscriber Screenshot

(Dividing the batch screenshot into 3 segments)


```
+--------------------+----+-------------------+-----+--------------------+-----+--------------------+------+--------------------+--------------------+
|         business_id|cool|               date|funny|           review_id|stars|                text|useful|             user_id|               words|
+--------------------+----+-------------------+-----+--------------------+-----+--------------------+------+--------------------+--------------------+
|-K82LBrI3H0FVuhTb...|   0|2013-08-18 22:24:19|    1|LEktDPyqSVdA3Psls...|  3.0|I'll keep my revi...|     1|_iX_y6ZgVmFQfs-hV...|[i, ll, keep, my,...|
|-K82LBrI3H0FVuhTb...|   0|2013-09-07 06:20:15|    0|5H3KETOvkbR4DuSsJ...|  4.0|I've been here a ...|     1|5LJAsGHJvLgGLX8s5...|[i, ve, been, her...|
|-K82LBrI3H0FVuhTb...|   0|2013-10-11 23:32:28|    1|RjakdZ4CqbMCE3-wI...|  4.0|Right next door t...|     1|67zOuDms6nAJDFXdO...|[right, next, doo...|
+--------------------+----+-------------------+-----+--------------------+-----+--------------------+------+--------------------+--------------------+
```



```
+--------------------+--------------------+--------------------+--------------------+--------------------+----------+
|            filtered|                cV|            features|       rawPrediction|         probability|prediction|
+--------------------+--------------------+--------------------+--------------------+--------------------+----------+
|[ll, keep, review...|(10000,[0,2,4,6,8...|(10002,[0,2,4,6,8...|[-11.547536869625...|[5.57825800108149...|       4.0|
|[ve, handful, tim...|(10000,[2,4,6,14,...|(10002,[2,4,6,14,...|[-11.547027335420...|[5.11636976820931...|       4.0|
|[right, next, doo...|(10000,[0,1,2,31,...|(10002,[0,1,2,31,...|[-11.547048582816...|[1.62566591457803...|       4.0|
+--------------------+--------------------+--------------------+--------------------+--------------------+----------+
```



```
+--------------------+
|Batch 171 accuracy|
+--------------------+
|0.6666666666666666|
+--------------------+
```


- Final Output for multiple batches

Latency = Time Elapsed / Count

Batch 6


```
+--------------------+------------------+----------------------+--------------+--------------------+
|  Batch 6 F1 Score|Batch 6 Accuracy|Batch 6 Time Elapsed|Batch 6 Count|  Batch 6 Latency|
+--------------------+------------------+----------------------+--------------+--------------------+
|0.7333333333333334|              0.75|    4.053088903427124|             4|1.013272225856781|
+--------------------+------------------+----------------------+--------------+--------------------+
```

Batch 43

Batch 43 F1 Score	Batch 43 Accuracy	Batch 43 Time Elapsed	Batch 43 Count	Batch 43 Latency
0.7777777777777777	0.6666666666666666	2.963904857635498	3	0.9879682858784994

Batch 74

Batch 74 F1 Score	Batch 74 Accuracy	Batch 74 Time Elapsed	Batch 74 Count	Batch 74 Latency
1.0	1.0	3.0699636936187744	3	1.023321231206258

Conclusion

- The python files for all the above mentioned tasks are as follows:
  - **Final_Code.py** : Training Models and Selecting the best one
  - **Test_Code.py** : To be used during demo on test JSON file
  - **new_producer.py** : Kafka Producer code for streaming data
  - **kafka_updated.py** : Subscriber code which loads model and calculates accuracy and latency
- The best model that was found out through batch computation task was Logistic Regression and the Pre-processing pipeline involved Tokenizer, Filtering and CountVectorizer
- Accuracy and F1-score was calculated for both the tasks – Batch Computation and Real-Time Computation
- Time taken for processing a batch was found to be proportional to the number of data points in that batch and latency for each data point in the batch comes out to be nearly the same