Project Motive / Question:

- This is an offical challenge by yelp where they given a chance for students to conduct research or analysis on our data and share their discoveries with us.
- Natural Language Processing & Sentiment Analysis
  What's in a review? Is it positive or negative? Yelp's reviews contain a lot of metadata that can be mined and used to infer meaning, business attributes, and sentiment.

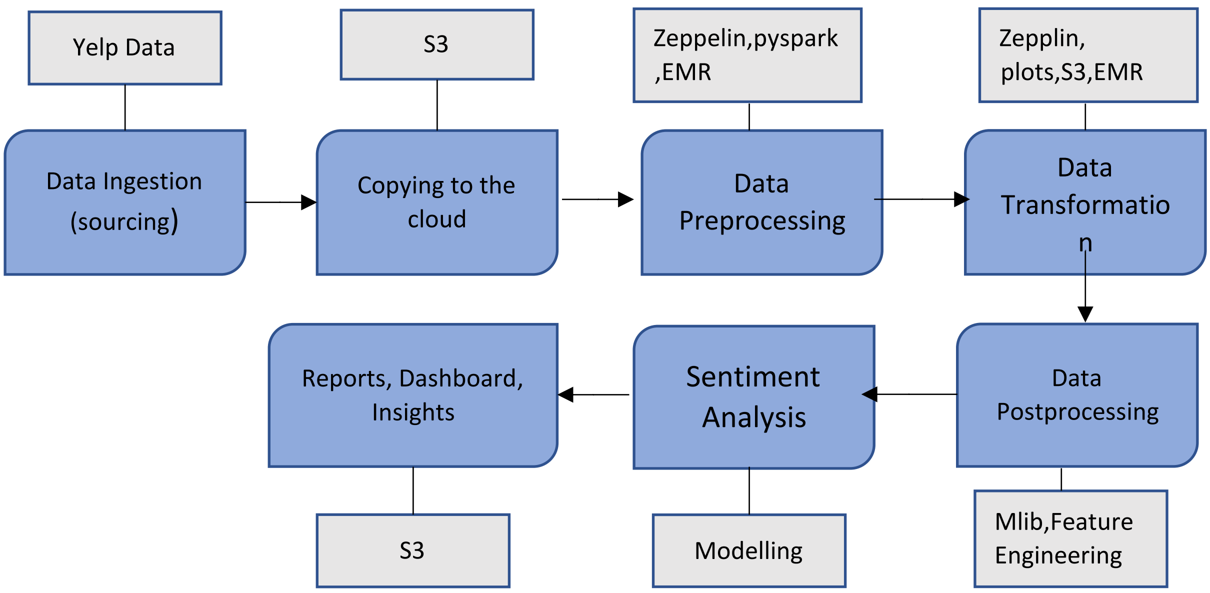
Proposed Approach: Tools / Techniques for:

- Amazon S3
- EC2
- EMR
- Apache Spark
- Spark MLIB
- Python
- Spark Graphx (Optinal)

Introduction:

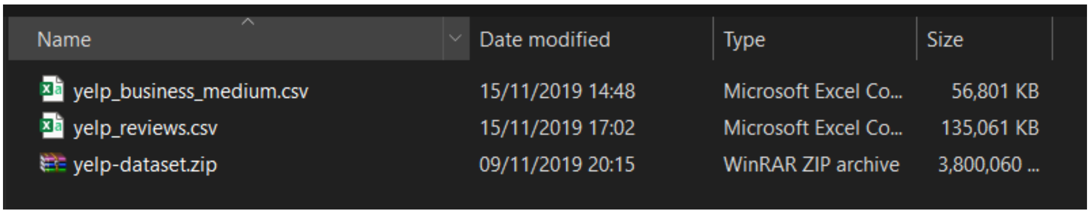
The project aims at predicting the user rating for a business based on sentiment analysis of the review given by the user (whether a user liked a local business or not). The rating prediction based on the user review will act as a performance parameter thereby, helping comparison of various businesses.

YELP DATA ANALYSIS PIPELINE/WORKFLOW



Step1 (Data Sourcing) :-

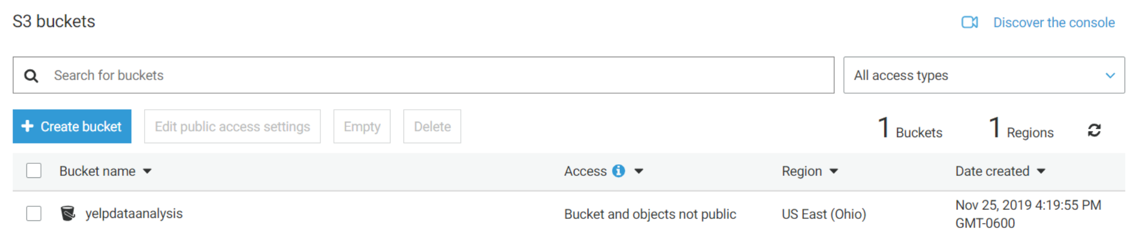
Download the source dataset from yelp website(section-4) to local disk and unzip it.

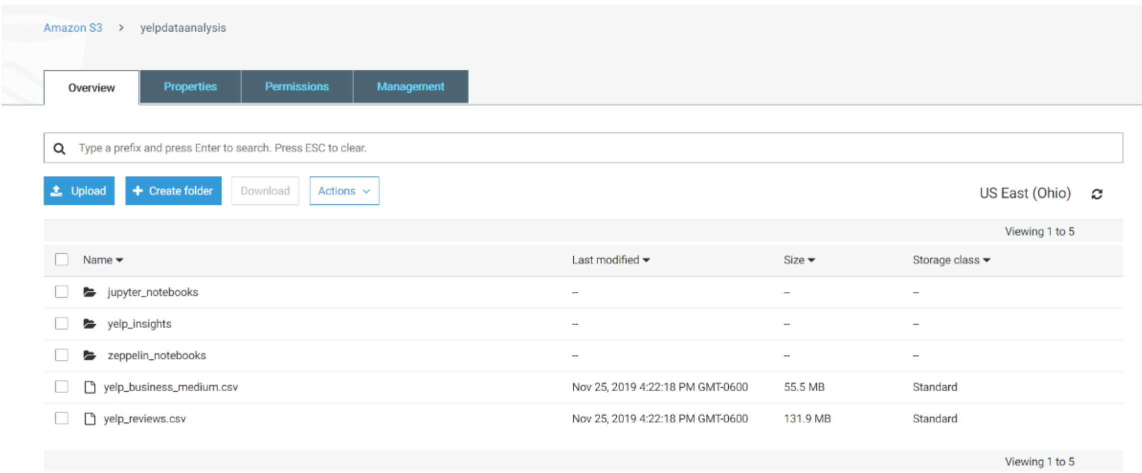


Step2 (Moving the data to the cloud):-

Copy the source datasets from Local Disk to Cloud environment (Amazon S3)

<u>Cost analysis:-</u>
$0.023 per gb - For first month
$0.115 (5 gb) – For the project





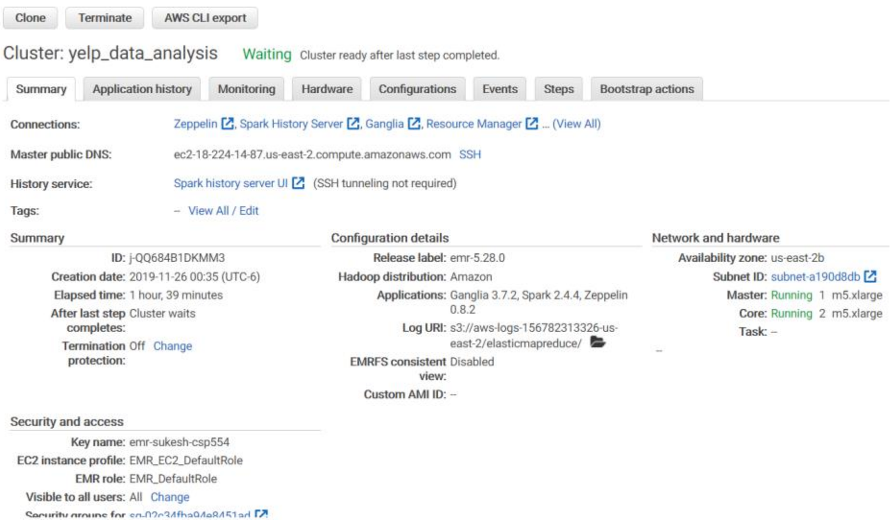
Step3 (Setting up the cloud envoirnment):-

Create the emr cluster with specific instance(virtual machine) considering your data size and cost parameter(for industrial purpose). Here I create m5.xlarge (4 vcpu's and 16 gb ram).

Cost Analysis:-

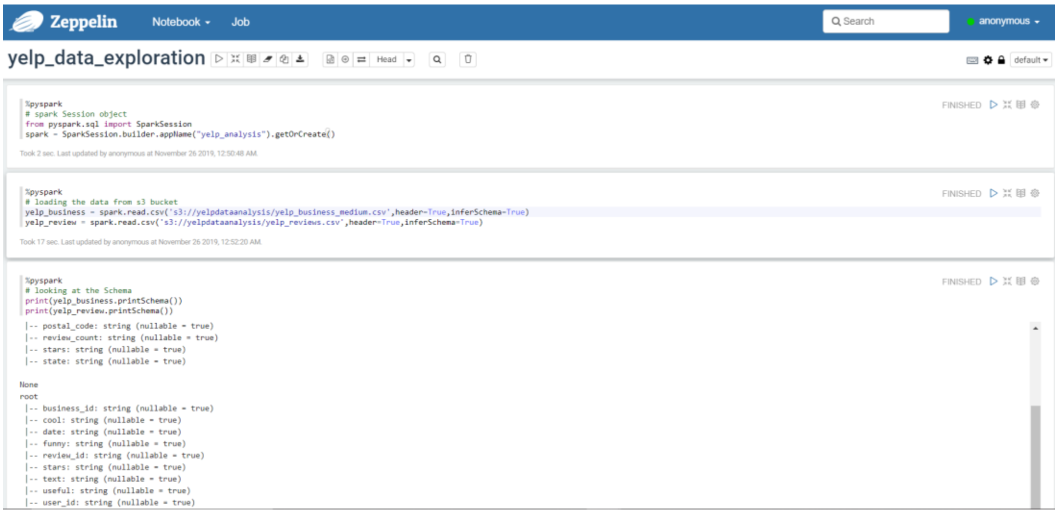
Emr_cluster_with_m5.xlarge = $0.192 per hour

For Project (8 hrs) = $1.52



Step4 (Data Preprocessing):-

We will start importing all the pyspark libraries requires in zeppelin notebook and create the spark session object with the specific confifurations (nodes/slaves) and look into the data and perform cleaning.



Step5 (Data exploration/analysis) :-

We can then perform our analysis and look some insights/patterns with help of visulizations and saving those insights, plots, requires data frames to the s3 bucket.

Sample code

```
%pyspark
#Top 10 category which has most business count

from pyspark.sql.functions import split,explode

category = yelp_business.select('categories')
individual_category = category.select(explode(split('categories', ',')).alias('category'))
grouped_category = individual_category.groupby('category').count()
top_category = grouped_category.sort('count',ascending=False)
print('Top 10 category which has most business count')
print(top_category.show(10,truncate=False))
```

```
Top 10 category which has most business count
+--------------------+-----+
|category            |count|
+--------------------+-----+
| 'validated': False |30162|
| Home Services      |6685 |
| Health & Medical   |4815 |
| Automotive         |4196 |
| Local Services     |3763 |
| Shopping           |3291 |
|Home Services       |2645 |
| Auto Repair        |2224 |
| Professional Services|2178 |
| 'classy': False    |2168 |
+--------------------+-----+
only showing top 10 rows

None
```

Took 2 sec. Last updated by anonymous at November 26 2019, 12:53:09 AM.

```
%pyspark
Top_ten_reviewed_business.select('name','categories','count').limit(10).write.save("s3://yelpdataanalysis/yelp_insights/top_reviewed_bussiness.csv", format='csv', header=True)
```

Took 15 sec. Last updated by anonymous at November 26 2019, 1:11:08 AM. (outdated)

Upload | Create folder | Download | Actions | US East (Ohio)

Name	Last modified	Size	Storage class
top_categories.csv	--	--	--
top_locations_who_have_number_of_bussiness.csv	--	--	--
top_ratings_given_by_user.csv	--	--	--
top_reviewed_bussiness.csv	--	--	--
best_category.png	Nov 26, 2019 2:22:27 AM GMT-0600	23.7 KB	Standard
most_reviews.png	Nov 26, 2019 2:22:27 AM GMT-0600	20.7 KB	Standard
star_rating_distributin.png	Nov 26, 2019 2:22:27 AM GMT-0600	14.4 KB	Standard

Viewing 1 to 7

Step6 (Data Postprocessing):-

For performing the sentiment analysis on the reviews, we need to convert the ratings attribute into binary class and tranform the data(features) in such a way for using sparkMlib. Here we used thresshold as > rating 4 for being postive review.

Sample code

```
%pyspark
# Removing the stop words and cleaning reviews
import string
import re

def remove_punct(text):
    regex = re.compile('[' + re.escape(string.punctuation) + '0-9\\r\\t\\n]')
    nopunct = regex.sub(" ", text)
    return nopunct

# giving a class label to the rating

def convert_rating(rating):
    if rating >=4:
        return 1
    else:
        return 0
```

```
+--------------------+-----+
|                text|stars|
+--------------------+-----+
|Total bill for th...|    0|
|I  adore  Travis ...|    1|
|I have to say tha...|    1|
|Went in for a lun...|    1|
| Today was my sec...|    0|
| I ll be the firs...|    1|
|Tracy dessert had...|    0|
|This place has go...|    0|
| I was really loo...|    0|
|It s a giant Best...|    0|
|Like walking back...|    1|
|Walked in around ...|    0|
|Wow  So surprised...|    1|
|Michael from Red ...|    1|
|I cannot believe ...|    0|
```

Step7 (Modeling/Sentiment Analysis):

While Feature engineering we did try the length of the words in a reviewas a attribute , but it isn't helping in classifiying the postive or negative from statistics. We did use tf-idf to convert the reviews into vectors and transform the data into required shape to perform machine learning (naive Baye's) on it and evaluating it with test data.

Sample code

```
# There isn't much Difference, hence it cannot be is used as attribute
```

```
+-----+-----------------+
|stars|      avg(length)|
+-----+-----------------+
|    0| 358.4106661671254|
|    1|295.06875408579606|
+-----+-----------------+
```

```
%pyspark
# Feature Transformations

from pyspark.ml.feature import Tokenizer,StopWordsRemover, CountVectorizer,IDF

tokenizer = Tokenizer(inputCol="text", outputCol="token_text")
stopremove = StopWordsRemover(inputCol='token_text',outputCol='stop_tokens')
count_vec = CountVectorizer(inputCol='stop_tokens',outputCol='c_vec')
idf = IDF(inputCol="c_vec", outputCol="tf_idf")
```

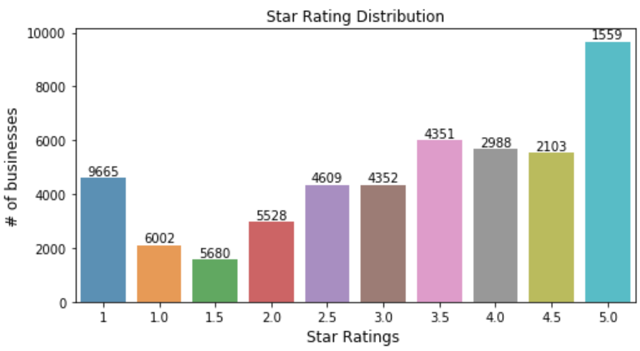
```
+-----+--------------------+
|label|            features|
+-----+--------------------+
|    0|(80152,[0,5,17,22...|
|    1|(80152,[0,3,5,18,...|
|    1|(80152,[0,3,8,12,...|
|    1|(80152,[0,25,29,3...|
|    0|(80152,[0,5,6,8,1...|
|    1|(80152,[0,1,4,11,...|
|    0|(80152,[2,9,24,90...|
|    0|(80152,[0,1,2,10,...|
|    0|(80152,[0,4,12,10...|
|    0|(80152,[0,2,8,14,...|
|    1|(80152,[0,6,7,10,...|
|    0|(80152,[0,4,5,14,...|
|    1|(80152,[0,9,15,20...|
|    1|(80152,[0,3,4,6,1...|
|    0|(80152,[0,5,7,8,1...|
```

```
%pyspark
acc_eval = MulticlassClassificationEvaluator()
acc = acc_eval.evaluate(test_results)
print("Accuracy of model at predicting positive or negative  was: {}".format(acc))
```

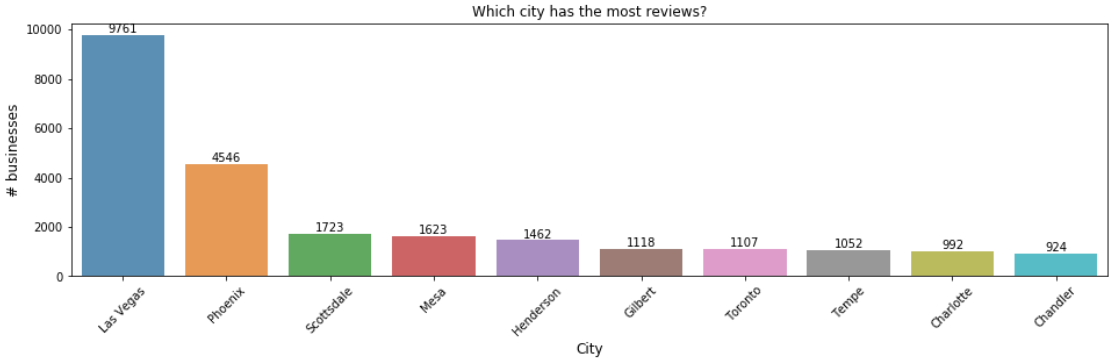
Results and Interpretation

Insights from data exploration

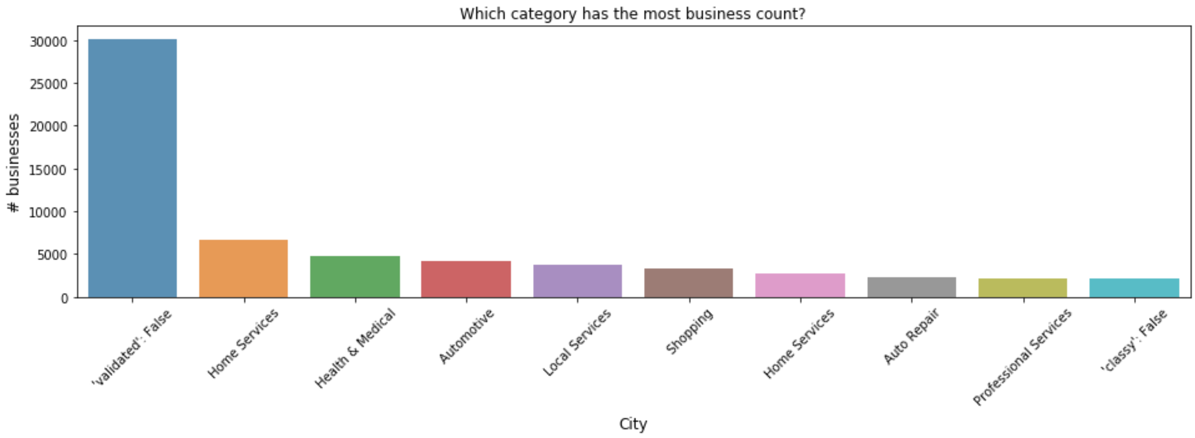
1.) Top ratings given by User to business



2.) Top Locations who have number of business more in world



3.) Top 10 category which has most bussiness count



Model analysis

```
%pyspark
# training the model
spam_predictor = nb.fit(training)

test_results = spam_predictor.transform(testing)

test_results.show()
```

```
+--------------------+-----+--------------------+--------------------+----------+
|            features|label|       rawPrediction|         probability|prediction|
+--------------------+-----+--------------------+--------------------+----------+
|       (80152,[],[])|    0|[-1.0737616963300...|[0.34172064734429...|       1.0|
|       (80152,[],[])|    0|[-1.0737616963300...|[0.34172064734429...|       1.0|
|       (80152,[],[])|    0|[-1.0737616963300...|[0.34172064734429...|       1.0|
|       (80152,[],[])|    0|[-1.0737616963300...|[0.34172064734429...|       1.0|
|(80152,[0,1,2,3,4...|    0|[-4404.0946723859...|[3.14169429702578...|       1.0|
|(80152,[0,1,2,3,4...|    0|[-1296.3175503156...|[0.99998646311423...|       0.0|
|(80152,[0,1,2,3,4...|    0|[-2951.8929448934...|[1.04825889292815...|       1.0|
|(80152,[0,1,2,3,4...|    0|[-3307.5982064010...|[2.31142993988170...|       1.0|
|(80152,[0,1,2,3,4...|    0|[-6069.3641568092...|[1.0,3.5550207656...|       0.0|
|(80152,[0,1,2,3,4...|    0|[-2513.7912741506...|[1.0,1.6502024249...|       0.0|
|(80152,[0,1,2,3,4...|    0|[-759.44892120455...|[0.99527012443285...|       0.0|
|(80152,[0,1,2,3,4...|    0|[-570.25508736558...|[0.00497429300961...|       1.0|
|(80152,[0,1,2,3,4...|    0|[-1867.8393489429...|[1.0,1.2670345678...|       0.0|
|(80152,[0,1,2,3,4...|    0|[-1599.4319706536...|[1.0,1.0931130918...|       0.0|
|(80152,[0,1,2,3,4...|    0|[-1084.9208174000...|[0.99999997732314...|       0.0|
```

```
%pyspark
acc_eval = MulticlassClassificationEvaluator()
acc = acc_eval.evaluate(test_results)
print("Accuracy of model at predicting positive or negative  was: {}".format(acc))
#Not bad considering we're using straight math on text data!
# We can Try switching out with multiple classification models!
# Or even try to come up with other engineered features!
```

```
Accuracy of model at predicting positive or negative  was: 0.817770737566
```

From the accuracy we can say , the model is way better than some random model and we can improve it by switching out with multiple classification models and also try to come up with more engineered features!

References:

- https://www.yelp.com/dataset/challenge
- https://www.kaggle.com/yelp-dataset/yelp-dataset
- https://spark.apache.org/docs/latest/sql-programming-guide.html
- https://hortonworks.com/apache/hdfs/
- https://changhsinlee.com/pyspark-dataframe-basics/
- https://monkeylearn.com/sentiment-analysis/
- https://www.edureka.co/community/12000/what-the-difference-betweenrdd-and-dataframes-apache-spark
- https://spark.apache.org/docs/latest/sql-programming-guide.html
- https://www.linkedin.com/pulse/choosing-machine-learning-frameworksapache-mahout-vs-debajani/