DATA MANAGEMENT, WAREHOUSING, AND ANALYTICS

CSCI 5408
Fall 2018

ASSIGNMENT 3

Gurjot Singh (B00811724)- gurjot@dal.ca

Devanshu Srivastava (B00810667)- dv960112@dal.ca

Task Description:

The objective is to learn, how to stream live data from social media platforms, obtain familiarity with apache spark, apply machine learning techniques, and use classification algorithms along with pattern recognition techniques on the tweets.

The first step was to create a personal account on a cloud hosting that offers Infrastructure as a service. IBM cloud(Watson Studio), Visual Studio and Jupyter Notebook were used in the development. Twitter account was setup and security credentials were fetched from Twitter Developer API.

The first program is developed in Apache Spark using Python language to stream real time Twitter Data into cloud platform. In this step we extracted 2000 tweets for performing classification. Next, we cleaned the live data to eliminate non-alphanumeric characters and blank spaces to perform Sentiment Analysis.

In the next step, we performed Sentiment Analysis on Twitter Data using a classifier. 'Tweets.csv' data is used as a label training data. The classifier used in this process was Logistic Regression.

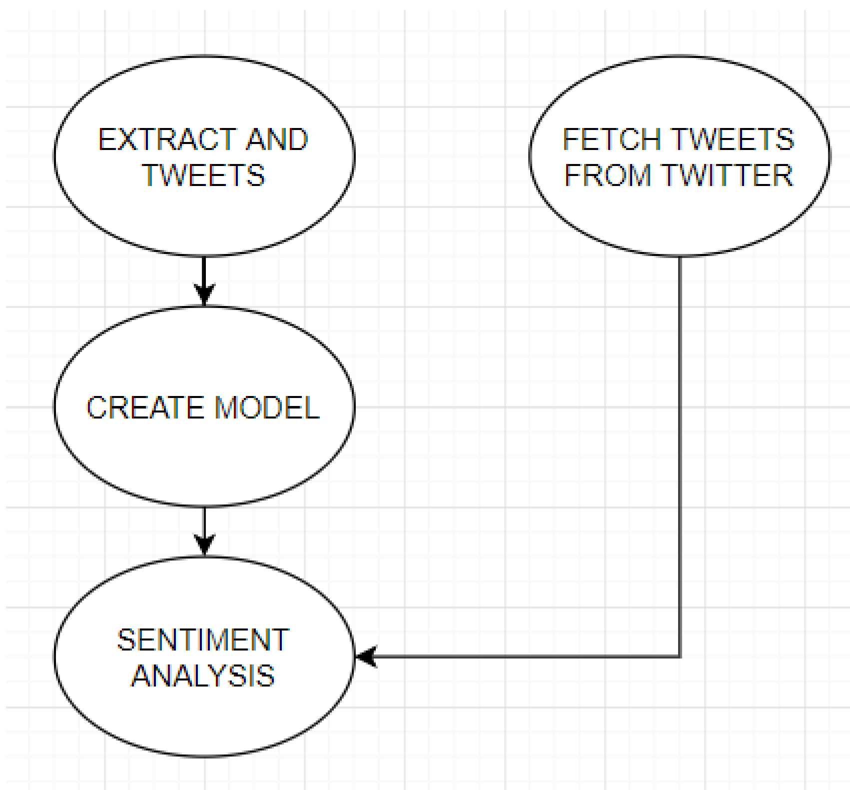


Fig1: Task Description

Tweet Extraction:

Tweets Extraction was done using Elastic Search[1]. The live streaming of data is performed using dynamic tweets and fetched using Elastic Search. The tweets were cleaned to enhance sentimental analysis process.

We fetched the "Tweets.csv" file, imported and uploaded it to a dataframe. We used Pandas in the process to extract data from the file. Also, Pandas were used to convert list to the dataset. The columns: 'text' and 'airline_sentiment' were extracted from the file which contain the tweets data and their sentiments respectively. The sentiments were calculated depending on three types of polarities: positive, negative, and neutral.

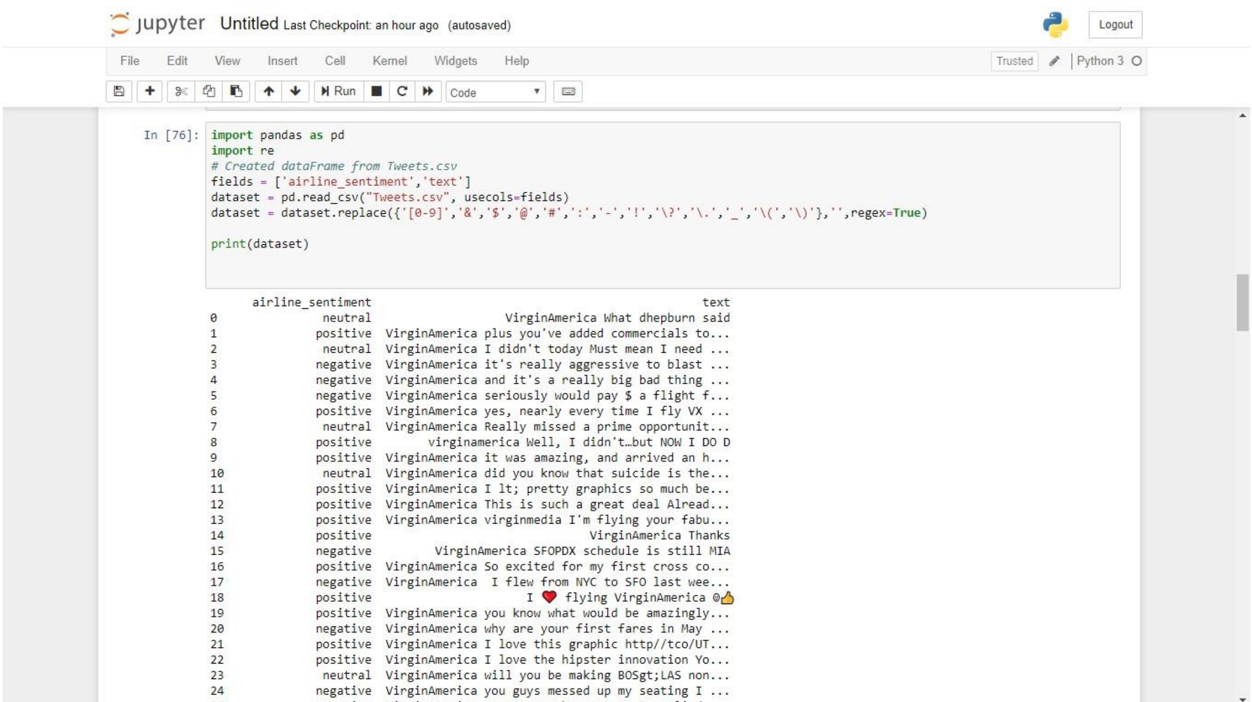


Fig 2: Creating and cleaning a Dataset

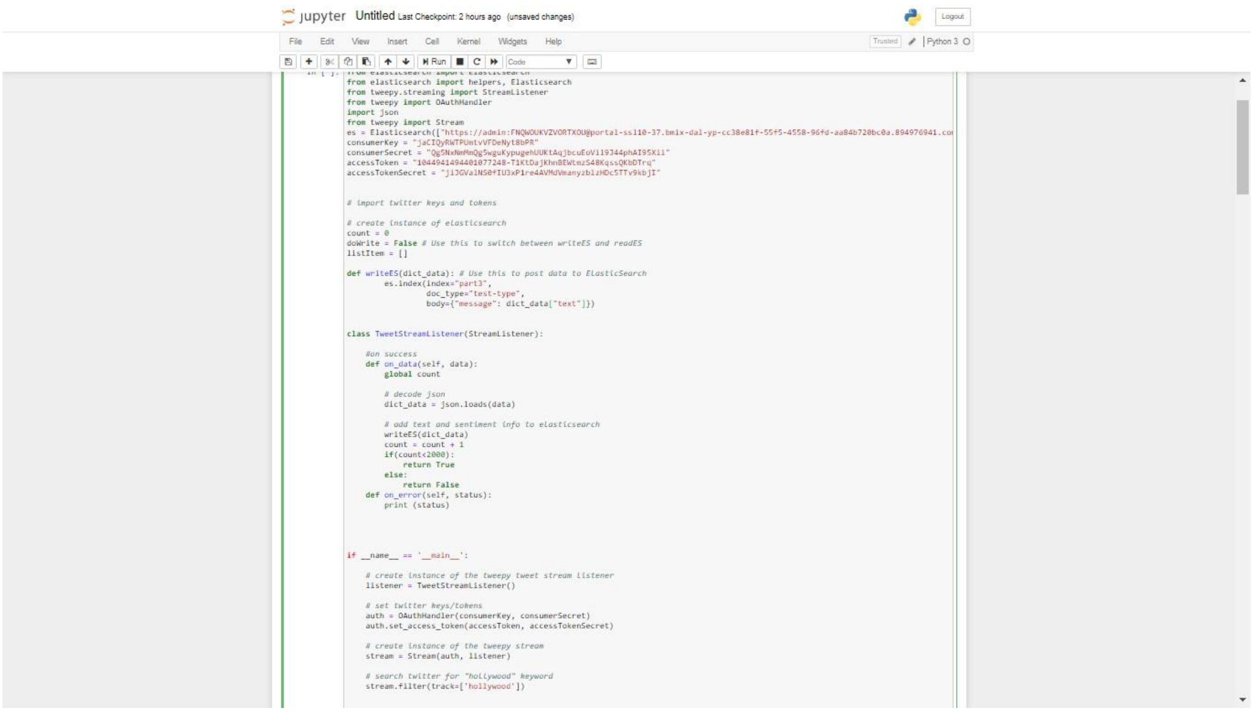


Fig 3: Fetching Data Using ElasticSearch

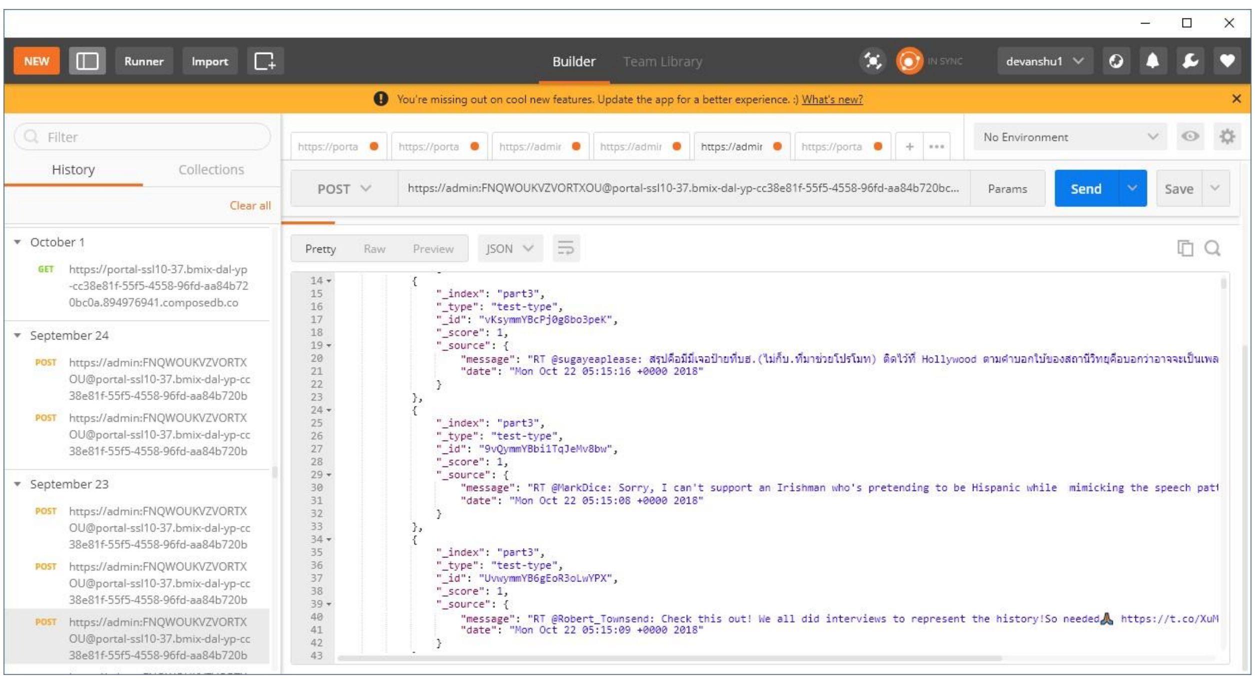


Fig 4: Tweets Fetched Using ElasticSearch

Sentiment Analysis:

To perform sentiment analysis, the live tweets data was loaded into a dataframe. This dataframe was used to perform analysis using Logistic Regression[2]. Using this, the generalization error is estimated and analyzed with a model. This is done by looking at the polarity in which logistic regression is predicted with a high probability for a positive sentiment. We found this classifier the most efficient and provided the results in a more efficient way in predicting the sentiments.

```
|         positive|VirginAmerica you...|  2.0|
+-----------------+--------------------+-----+
only showing top 20 rows

PipelineModel_4bee9bcd157f62a029ed
```

```
In [24]: from elasticsearch import Elasticsearch
from elasticsearch import helpers, Elasticsearch
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
import json
from tweepy import Stream
es = Elasticsearch(["https://admin:FNQWOUKVZVORTXOU@portal-ssl10-37.bmix-dal-yp-cc38e81f-55f5-4558-96fd-aa84b720bc0a.894976941.co
consumerKey = "jaCIQyRWTPUmtvVFDeNyt8bPR"
consumerSecret = "Qg5NxNmMmQg5wguKypugehUUKtAqjbcuEoVil9J44phAI95Xii"
accessToken = "1044941494401077248-T1KtDajKhnBEWtmzS48KqssQKbDTrq"
accessTokenSecret = "jiJGValNS0fIU3xP1re4AVMdVmanyzblzHDc5TTv9kbjI"
def readES(): # Use this to get data from ElasticSearch
        listItem=[]
        res = es.search(index="part3", body={},size=2000, from_=0)
        for hit in res['hits']['hits']:
            listItem.append(hit['_source']['message'])
          # // print("%(message)s" % hit["_source"])
        return pd.DataFrame(listItem)
dFrame =  readES()
dFrame = dFrame.replace({'[0-9]','&','$','@','#',':','-','!','\?','\.','_','\(','\)'},'',regex=True)
finalDF = spark.createDataFrame((dFrame),['text'])

prediction1 = model.transform(finalDF)
# evaluator = BinaryClassificationEvaluator(rawPredictionCol="text",labelCol="Out")
# evaluator.evaluate(prediction)
selected = prediction1.show(2000)
```

```
+--------------------+--------------------+--------------------+--------------------+--------------------+-------------------
-+----------+
|                text|                word|         outFeatures|            features|       rawPrediction|        probabilit
y|prediction|
+--------------------+--------------------+--------------------+--------------------+--------------------+-------------------
-+----------+
```

Fig 5: Final Output Program

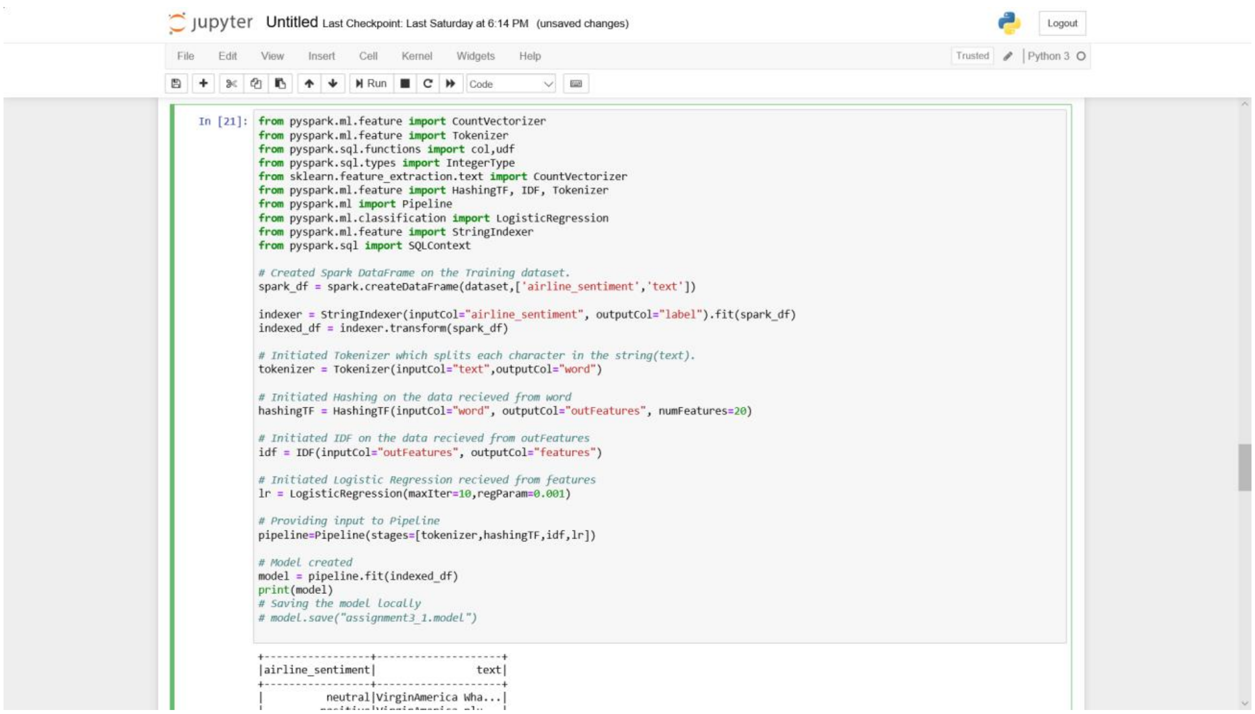
```
selected = prediction1.show(2000)
```

```
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
----------+
|                text|                word|         outFeatures|            features|       rawPrediction|         probability|
prediction|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
----------+
|RT sugayeaplease ...|[rt, sugayeapleas...|(20,[2,12,13],[2....|(20,[2,12,13],[1....|[0.19013367091943...|[0.39756738668861...|
0.0|
|RT MarkDice Sorry...|[rt, markdice, so...|(20,[0,1,3,4,7,8,...|(20,[0,1,3,4,7,8,...|[1.78350614556806...|[0.85023479400251...|
0.0|
|RT RobertTownsend...|[rt, roberttownse...|(20,[0,2,3,7,8,10...|(20,[0,2,3,7,8,10...|[0.59381070621750...|[0.54908855596086...|
0.0|
|RT SHOTALOVE Holl...|[rt, shotalove, h...|(20,[3,8,12,13,14...|(20,[3,8,12,13,14...|[0.46958879737617...|[0.48294936087689...|
0.0|
|RT fedhr Añadiero...|[rt, fedhr, añadi...|(20,[1,2,3,4,5,10...|(20,[1,2,3,4,5,10...|[1.63208027400494...|[0.85176092549009...|
0.0|
|RT badposthisisus...|[rt, badposthisis...|(20,[1,2,3,5,7,10...|(20,[1,2,3,5,7,10...|[1.22607194657269...|[0.72686864513305...|
0.0|
|RT ItalianMonster...|[rt, italianmonst...|(20,[0,2,4,6,7,8,...|(20,[0,2,4,6,7,8,...|[1.68576209671985...|[0.86228821193220...|
0.0|
|RT gmxn The most ...|[rt, gmxn, the, m...|(20,[1,2,4,5,6,8,...|(20,[1,2,4,5,6,8,...|[1.05981989900158...|[0.69227552865625...|
0.0|
|RT WyattEarpLA Ob...|[rt, wyattearpla,...|(20,[0,1,3,7,8,11...|(20,[0,1,3,7,8,11...|[1.63132030065404...|[0.80158814139876...|
0.0|
|RT SHOTALOVE Holl...|[rt, shotalove, h...|(20,[3,8,12,13,14...|(20,[3,8,12,13,14...|[0.46958879737617...|[0.48294936087689...|
0.0|
|RT SHOTALOVE Holl...|[rt, shotalove, h...|(20,[3,8,12,13,14...|(20,[3,8,12,13,14...|[0.46958879737617...|[0.48294936087689...|
0.0|
|Selma Blair Has R...|[selma, blair, ha...|(20,[0,2,3,9,10,1...|(20,[0,2,3,9,10,1...|[0.20626650299721...|[0.40401212822174...|
0.0|
|RT SHOTALOVE Holl...|[rt, shotalove, h...|(20,[3,8,12,13,14...|(20,[3,8,12,13,14...|[0.46958879737617...|[0.48294936087689...|
0.0|
|RT elenajcl RealJ...|[rt, elenajcl, re...|(20,[1,3,5,6,8,9,...|(20,[1,3,5,6,8,9,...|[2.70405540236529...|[0.94585320661141...|
0.0|
|"L Ron Hubbard "S...|["l, ron, hubbard...|(20,[0,1,3,4,5,6,...|(20,[0,1,3,4,5,6,...|[1.15703278531926...|[0.73177083291683...|
0.0|
|RT MinSugaPeru 🎵...|[rt, minsugaperu,...|(20,[1,3,4,5,9,12...|(20,[1,3,4,5,9,12...|[2.36364045442457...|[0.91487649676531...
|       0.0|
```

Fig 6: Final Output after Analysis

Training Data:

'Tweets.csv' is our training dataset and we created spark data frame on it. We imported tokenizer and used the tokens to split the tweets to individual words, and calculate the sentiment score for it. Next, we used TF-IDF(Term Frequency- Inverse Data Frequency to extract the feature of tweets[3]. The main functionality of TF is to calculate the frequency of each term. It is defined as the ratio of the number of times a word exists in a document to the total number of words in that document. Using IDF method we calculated the weight of each word in the tweet[4].




```python
from pyspark.ml.feature import CountVectorizer
from pyspark.ml.feature import Tokenizer
from pyspark.sql.functions import col,udf
from pyspark.sql.types import IntegerType
from sklearn.feature_extraction.text import CountVectorizer
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import StringIndexer
from pyspark.sql import SQLContext

# Created Spark DataFrame on the Training dataset.
spark_df = spark.createDataFrame(dataset,['airline_sentiment','text'])

indexer = StringIndexer(inputCol="airline_sentiment", outputCol="label").fit(spark_df)
indexed_df = indexer.transform(spark_df)

# Initiated Tokenizer which splits each character in the string(text).
tokenizer = Tokenizer(inputCol="text",outputCol="word")

# Initiated Hashing on the data recieved from word
hashingTF = HashingTF(inputCol="word", outputCol="outFeatures", numFeatures=20)

# Initiated IDF on the data recieved from outFeatures
idf = IDF(inputCol="outFeatures", outputCol="features")

# Initiated Logistic Regression recieved from features
lr = LogisticRegression(maxIter=10,regParam=0.001)

# Providing input to Pipeline
pipeline=Pipeline(stages=[tokenizer,hashingTF,idf,lr])

# Model created
model = pipeline.fit(indexed_df)
print(model)
# Saving the model locally
# model.save("assignment3_1.model")
```

```
+-----------------+--------------------+
|airline_sentiment|                text|
+-----------------+--------------------+
|          neutral|VirginAmerica Wha...|
```

Fig 7: Model Creation in Python

Discussion

Sentimental Analysis using Apache Spark increased the speed and efficiency of implementing the analysis on the Tweets Data. Sentimental Analysis performed earlier took more time when executing the programs, whereas now since the data is on the cloud, the results were provided in a very short time.

Also, with the use of Spark, the integration of the database and the tweets extraction was much more easier and realistic. The fundamentals of the spark i.e. the dataframes made the tweets extraction and execution process more simple.

Executing 2000 tweets on a local machine without Apache Spark would have been extracted in hours of time whereas using Apache Spark, the execution was completed in a couple of seconds and large data was retrieved.

Code Submission

The code has been submitted to a source code repository. Access have been given to tmeredith, shilpa, lyadav, prajapati, and psachdeva. The link for the repository is :

https://git.cs.dal.ca/devanshu/Apache_Spark_SentimentAnalysis.git

References

[1]"How to Query Elasticsearch with Python", *Marco Bonzanini*, 2018. [Online]. Available: https://marcobonzanini.com/2015/02/02/how-to-query-elasticsearch-with-python/. [Accessed: 18- Oct- 2018].

[2]"Classification and regression - Spark 2.1.0 Documentation", *Spark.apache.org*, 2018. [Online]. Available: https://spark.apache.org/docs/2.1.0/ml-classification-regression.html. [Accessed: 18- Oct- 2018].

[3]"Lab: Introduction to Spark ML: Sentiment Analysis - IST718", *Classes.ischool.syr.edu*, 2018. [Online]. Available: http://classes.ischool.syr.edu/ist718/content/unit09/lab-sentiment_analysis/. [Accessed: 09- Oct- 2018].

[4]"tthustla/setiment_analysis_pyspark", *GitHub*, 2018. [Online]. Available: https://github.com/tthustla/setiment_analysis_pyspark/blob/master/Sentiment%20Analysis%20with%20PySpark.ipynb. [Accessed: 14- Oct- 2018].