CS4830: Big Data Laboratory

Final Project Report

Rahul Chakwate (AE16B005)
S.Durga Sandeep (ME16B125)
Shiv Tavker (ME16B128)

September 7, 2020

Contents

1 Introduction **2**

2 Exploratory Data Analysis **2**
2.1 Top 20 rows: . . . 2
2.2 Schema of the dataset: . . . 3
2.3 Top 10 users: . . . 4
2.4 Average number of reviews per user and Cumulative Distribution of User reviews: . . . 4
2.5 Imbalance in the Ratings: . . . 5
2.6 Mix of Languages English + Other Languages: . . . 5

3 Data Pre-processing: **6**
3.1 Tokenisation: . . . 6
3.2 Stopwords Removal: . . . 6
3.3 Normalisation: . . . 6

4 Feature Engineering Techniques: **7**
4.1 Count Vectorizer: . . . 7
4.2 TF-IDF features: . . . 7
4.3 N-Grams: . . . 7
4.4 # Characters and # Words in the Sentence: . . . 7

5 Modelling Techniques: **7**
5.1 Naive Bayes: . . . 7
5.2 Logistic Regression: . . . 9
5.3 Random Forest: . . . 11

6 Real-time Computation **12**
6.1 Streaming: . . . 12
6.2 Latency Calculation: . . . 13

1 Introduction

Yelp dataset consists of business, check-ins and review datasets. In this project, we are going to explore on the review dataset. Given the review dataset, we can perform different tasks like sentiment analysis (which is our task), word embedding training particular to reviews, analysing the user behaviour for better marketing strategies and several other applications specific to your objective.

Here in this project we are predicting the user rating from 1 to 5 based on the given text features, date-timestamp, and other numerical features. Basically this can be treated as Multi-class classification or a Regression task. We applied several machine learning models like Naive bayes, Logistic regression with polynomial degree, Random forest on the dataset. Finalised the naïve bayes model based on cross-validation score. Interestingly, we can apply a regression model on this dataset. This is because the fact that classes are ordinal i.e they have specific order among the ratings which are $5 > 4 > 3 > 2 > 1$ unlike simple nominal classification.

But cons of this approach is we need to do extensive validation on deciding the threshold as dataset is imbalanced (We could not finish this approach because of credit issues at the end, but our classification model performed decent enough)

2 Exploratory Data Analysis

Note: We visualised only 50% of the data. All observations are based on this sample dataset Each review is accompanied by a rating from 1 to 5 stars. I will remove all columns from the dataset apart from the review text itself, the date, funny and the star rating which will be used to predict the positive/negative sentiment. Let's now visually inspect some of the data.

2.1 Top 20 rows:

Let's look at the top 20 rows of the dataset, and schema of the dataset to know what kind of features are available:

business_id	cool	date	funny	review_id	stars	text	useful	user_id	length	words	filtered
-MhfebM0QIsKt87iD...	0	2015-04-15 05:21:16	0	xQY8N_XvtGbearJ5X...	2.0	As someone who ha...	5	OwjRMXRC0KyPrIlcj...	1373	[as, someone, who...	[someone, worked,...
1brU8StCq3yDfr-QM...	0	2013-12-07 03:16:52	1	UmFMZ8PyXZTY2Qcwz...	1.0	I am actually hor...	1	nIJD_7ZXHq-FX8byP...	1407	[i, am, actually,...	[actually, horrif...
HQl28KMwrEKHqhFrr...	0	2015-12-05 03:18:11	0	LG2ZaYiOgpr2DK_90...	5.0	I love Deagan's. ...	1	V34qejxNsCbcgD8C0...	431	[i, love, deagan'...	[love, deagan's.,...
5JxlZaqCnk1MnbgRi...	0	2011-05-27 05:30:52	0	i6g_oA9Yf9Y31qt0w...	1.0	Dismal, lukewarm,...	0	ofKDkJKXSKZXu5xJN...	401	[dismal,, lukewar...	[dismal,, lukewar...
IS4cv902ykd8wj1TR...	0	2017-01-14 21:56:57	0	6TdNDKywdbjoTkize...	4.0	Oh happy day, fin...	0	UgMW8bLE0QMJDCkQ1...	841	[oh, happy, day,,...	[oh, happy, day,,...
nlxHRv1zXGT0c0K5l...	0	2013-05-07 07:25:25	0	L2O_INwlrRuoX05KS...	5.0	This is definitel...	2	5vD2kmE25YBrbayKh...	523	[this, is, defini...	[definitely, favo...
Pthe4qk5xh4n-ef-9...	0	2015-11-05 23:11:05	0	ZayJ1zWyWgY9S_TRL...	5.0	Really good place...	1	aq_ZxGHir48TUXJ1...	341	[really, good, pl...	[really, good, pl...
FNCJpSn0tL9iqoY3J...	0	2017-07-18 18:31:54	0	lpFIJYpsvDxyph-kP...	5.0	Awesome office an...	0	dsd-KNYKMpx6ma_sR...	237	[awesome, office,...	[awesome, office,...
e_BiI4ej1CWlF0EyV...	0	2015-02-16 06:48:47	0	JA-xnyHytKiOIHl_z...	5.0	Most delicious au...	0	P6apihD4ASf1vpPxH...	216	[most, delicious,...	[delicious, authe...
Ws8V970-mQt2X9CwC...	1	2009-10-13 04:16:41	0	z4BCgTkfNtCu4XY5L...	4.0	I have been here ...	3	jOERvhmK6_lo_XGUB...	686	[i, have, been, h...	[twice., nice, la...
PA61RWk3AMwOEXHev...	0	2013-12-28 21:02:55	0	TfVth7UNfgilv4J3e...	5.0	Maria is VERY goo...	0	s5j_CRBWDCCMDJ6r7...	504	[maria, is, very,...	[maria, good, gre...
l-nL48mhzpZjcavoo...	1	2015-10-17 01:38:13	1	Tyx7AxYQfSRnBFUIX...	4.0	ORDER In <Deliver...	2	HJECayULRM-6xh2GC...	741	[order, in, <deli...	[order, <delivery...
Naa6E0YU0Wr7jCuCE...	0	2015-07-03 21:48:51	0	wJMjt5C2ylRKgY0Xb...	5.0	We purchased new ...	3	lYIQGP-a534nyksaw...	447	[we, purchased, n...	[purchased, new, ...
Ns4tjgLfqRlqawGlN...	0	2016-06-11 22:00:11	0	QCxPzh7cuxJrLd6A_...	5.0	Everything that m...	0	qftVgPj_kRTildMDj...	475	[everything, that...	[everything, husb...
ZlCSsWS07JulSBIQl...	0	2015-05-26 10:36:47	0	qWHp2l2lysENZObh6...	5.0	Called for a 5:15...	1	5lb0POg2t-AkMFx66...	246	[called, for, a, ...	[called, 5:15, mo...
7Ka9Pd8X9SRHs1D5E...	0	2017-08-07 21:36:36	0	mjbs5CL4eMu4o6_Vt...	1.0	If I could give l...	0	TF4C-F5iqavACQgKT...	936	[if, i, could, gi...	[give, less, one,...
d4qwVw4PcN-_2nK2o...	0	2015-02-02 06:28:00	0	bVTjZgRNq8Toxzvti...	1.0	10pm on a super b...	0	2hRe26HSCAWbFRn5W...	102	[10pm, on, a, sup...	[10pm, super, bow...
oVuZtlCFg_zF090Nh...	1	2018-02-01 19:15:00	0	Ne_2CSfcKIqXHmw_K...	4.0	A close friend wa...	3	6sJN_HlM_uwpfLJ1p...	1897	[a, close, friend...	[close, friend, t...
_iGvLfEsqDwPUxRUA...	0	2017-06-28 00:39:18	0	Hy-gUXQh3RVhE8FLH...	1.0	Tried to have my ...	0	kMkWON2lnw0s-M-fw...	541	[tried, to, have,...	[tried, car, repa...
poSV39UqEg-gpESXa...	0	2018-03-04 01:03:53	0	UGErdm6bt48SXTVwJ...	3.0	My husband and I ...	0	QodunSzok4nIYFNrT...	714	[my, husband, and...	[husband, go, wee...

Figure 1

2.2 Schema of the dataset:

Field name	Type	Mode
text	STRING	NULLABLE
cool	INTEGER	NULLABLE
funny	INTEGER	NULLABLE
stars	FLOAT	NULLABLE
date	TIMESTAMP	NULLABLE
business_id	STRING	NULLABLE
user_id	STRING	NULLABLE
useful	INTEGER	NULLABLE
review_id	STRING	NULLABLE

Figure 2

As you can see the schema, we have stars as our label and remaining as our features but are in string format. So we need to encode the text features into the numerical values using Natural Language Processing Techniques.

Note:

1. Features like cool and funny have negatives which are not accepted in the Naïve Bayes (Our Final Model). So we won't be using those negative features for the prediction task.
2. Time-stamp – Feature extraction techniques like day, month, year, isWeekend, Week number, Quarter number and so-on can be done through datetime feature
3. Given business_id has unique values for each record, seems they are already hashed like user_id
4. User_id is not unique for every record, so there is repetitions in the user_id and we will explore it in the next section

2.3 Top 10 users:

	review_id	date		useful	funny	cool	stars
	count	min	max	sum	sum	sum	mean
user_id							
CxDOIDnH8gp9KXzpBHJYXw	3569	2009-11-09	2017-12-05	11345	5546	6695	3.201737
bLbSNkLggFnqwNNzzq-Ijw	2077	2012-05-20	2017-12-11	23572	12630	16303	3.256139
PKEzKWv_FktMm2mGPjwd0Q	1611	2008-12-12	2017-12-05	12078	4987	8183	3.664804
DK57YibC5ShBmqQl97CKog	1463	2006-02-12	2017-12-10	13175	6693	10795	3.826384
QJI9OSEn6ujRCtrX06vs1w	1322	2007-03-20	2016-05-01	8748	4104	6142	3.622542
d_TBs6J3twMy9GChqUEXkg	1184	2010-12-27	2017-11-23	2992	1341	1792	3.463682
ELcQDlf69kb-ihJfxZyL0A	1159	2011-04-10	2017-12-10	2155	1298	1029	2.993097
cMEtAiW60I5wE_vLfTxoJQ	1126	2009-06-12	2017-12-07	1943	1213	1509	4.066607
hWDybu_KvYLSdEFzGmiTw	1117	2009-03-08	2017-11-26	7234	4968	5762	3.640107
U4INQZOPSUaj8hMjLlZ3KA	1101	2008-05-13	2017-12-11	6116	2832	4187	3.714805

Figure 3

As you can see above, A user is preferred based on the review count. Besides we can see those 10 users statistics like minimum and maximum date of the review, and sum of their useful, funny, cool, average stars.

2.4 Average number of reviews per user and Cumulative Distribution of User reviews:

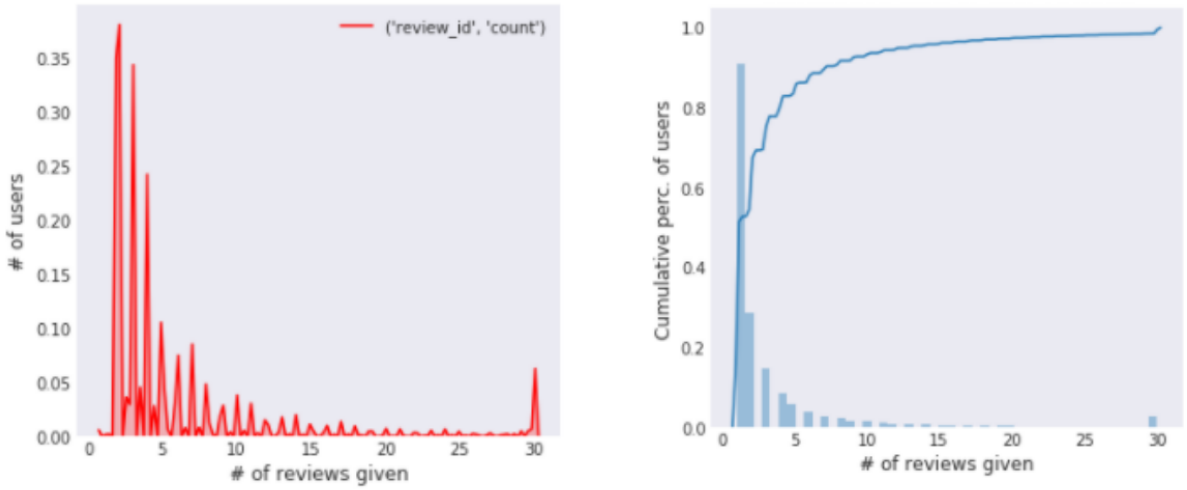


Figure 4

$\approx 80\%$ of the user gives 5 reviews, Maximum number of reviews is close to 30.

2.5 Imbalance in the Ratings:

We notice that 5-star reviews are the most popular (Majority class), and also that 1-star reviews are more common than 2- or 3-star reviews. We can assume that customers will go through the trouble of leaving a review only if they were highly impressed or highly disappointed. So order from the following histogram is, # 5 stars > # 4 stars > # 1 stars > # 3 stars > # 2 stars

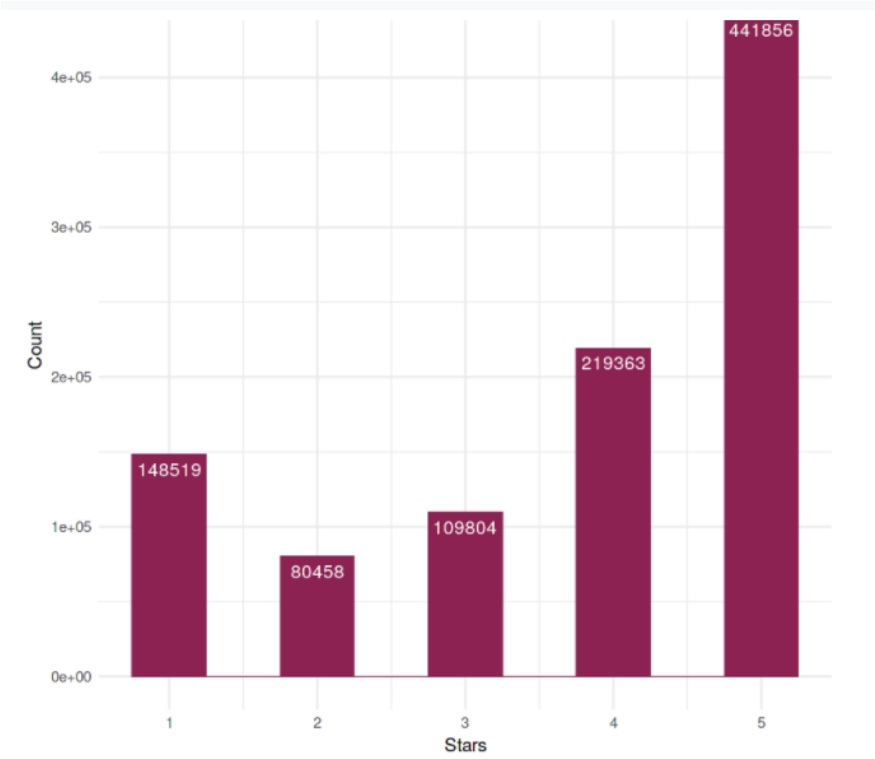


Figure 5

2.6 Mix of Languages English + Other Languages:

We used Fasttext (Open source library by Facebook) as a language detection model. The model identified around 80% to be english, and other languages like french, spanish are also found in the dataset Look at the following histogram. As you can see, English is the majority language.

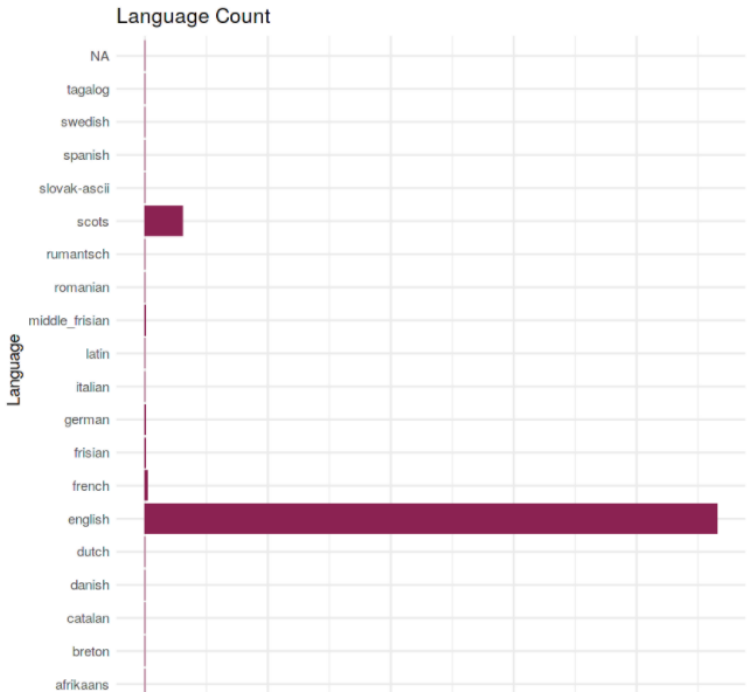


Figure 6

3 Data Pre-processing:

3.1 Tokenisation:

Tokenization is the process of converting text into tokens before transforming it into vectors. It is also easier to filter out unnecessary tokens. For example, a document into paragraphs or sentences into words. In this case we are tokenizing the reviews into words.

3.2 Stopwords Removal:

Stop words are the most commonly occuring words which are not relevant in the context of the data and do not contribute any deeper meaning to the phrase. In this case it contains no sentiment.

3.3 Normalisation:

Includes casing, Stemming/Lemmatisation Lower case all the characters for uniformity in the dataset. This is important when using a bag of words models. This process finds the base or dictionary form of the word known as the lemma. This is done through the use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations). This normalization is similar to stemming but takes into account the context of the word.

Note: Unfortunately, we could not find the lemmatization in the pyspark documents. So we skipped rhis part. But this had an impact on the final results of accuracy from the accuracy we got from the colab notebook.

4 Feature Engineering Techniques:

4.1 Count Vectorizer:

Here our aim is to help convert a collection of text reviews to vectors of token counts. When an a-priori dictionary is not available, CountVectorizer can be used as an Estimator to extract the vocabulary, and generates a CountVectorizerModel. The model produces sparse representations for the reviews over the vocabulary, which can then be passed to other algorithms. But cons of this technique is that order of the words is not taken into consideration, only the frequency of the words information has been stored. And also this gives high priority for highly frequent words, sometimes highly frequent words across all the reviews does not help in the sentiment analysis. So we apply tfidf feature extraction technique to avoid this issue and n-gram model is used to solve the order of words issue

4.2 TF-IDF features:

Term Frequency Inverse Document Frequency is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus aka reviews in our dataset. This method is used to encode the text features into numerical features. This technique is used to reduce the importance for the stopwords and increase relative importance to the unique words which helps in differentiating the reviews in the sentiment analysis.

4.3 N-Grams:

An n-gram is a sequence of n tokens (typically words) for some integer n. This is also like tokenisation but we split the sentence after every n words. This helps in preserving the order of the words in sentence to some extent.

4.4 # Characters and # Words in the Sentence:

This feature helped us in improving the cross-validation accuracy by 1%. We wrote a UDF function for this feature engineering technique. Refer to the code attached with this report for more details

5 Modelling Techniques:

We attempted different machine learning classification models to decide the best model suitable for this task. Specifically, we tried Naive Bayes, Logistic Regression and Random Forest Classifier models. We also tried Gradient Boosting Machines and Linear Support Vector Classifier with OneVsRest classifier but both the models took long to converge. Hence, we could not report their performance.

5.1 Naive Bayes:

Parameters:

- smoothing=1.0,
- modelType='multinomial'

For the Naive Bayes Classifier, we tried different variations of vectorizer and normalization in order to further improve the performance.

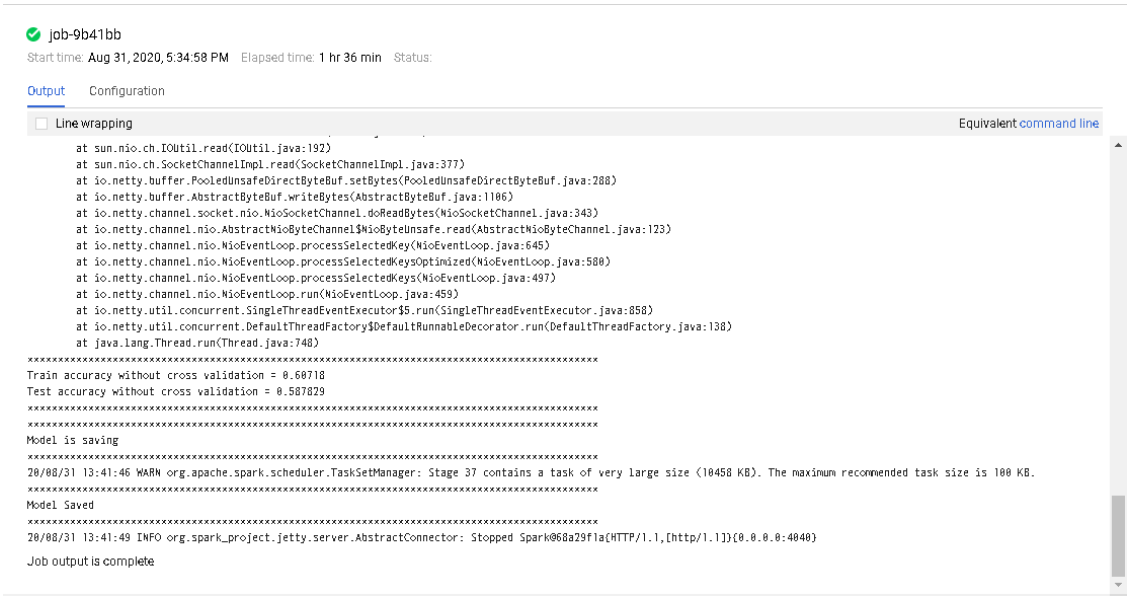


Figure 7: Naive Baves without l2-Normalization

Figure 8: Naive Bayes with Hashing TF

Figure 9: Naive Bayes with Count Vectorizer

Observations:

- We first implemented Naive Bayes Model with Count Vectorizer and Idf. With this we achieved 58.7% accuracy.
- However, we figured out a discrepancy between TfIdf Vectorizer of pyspark and sklearn. Sklearn's code uses l2-norm of Idf vectors which provides better features to the classifier. Hence, we implemented custom TfIdf with l2-norm which boosted the accuracy to around 63.1%.
- Further using count vectorizer instead of hashing TF gives a slight performance boost. However, hashing TF is faster.

5.2 Logistic Regression:

Following settings were used for Logistic Regression Model

Parameters:

- maxIter=10,
- regParam=0.3,
- elasticNetParam=0.8,
- family="multinomial"

Figure 10: Logistic Regression baseline model

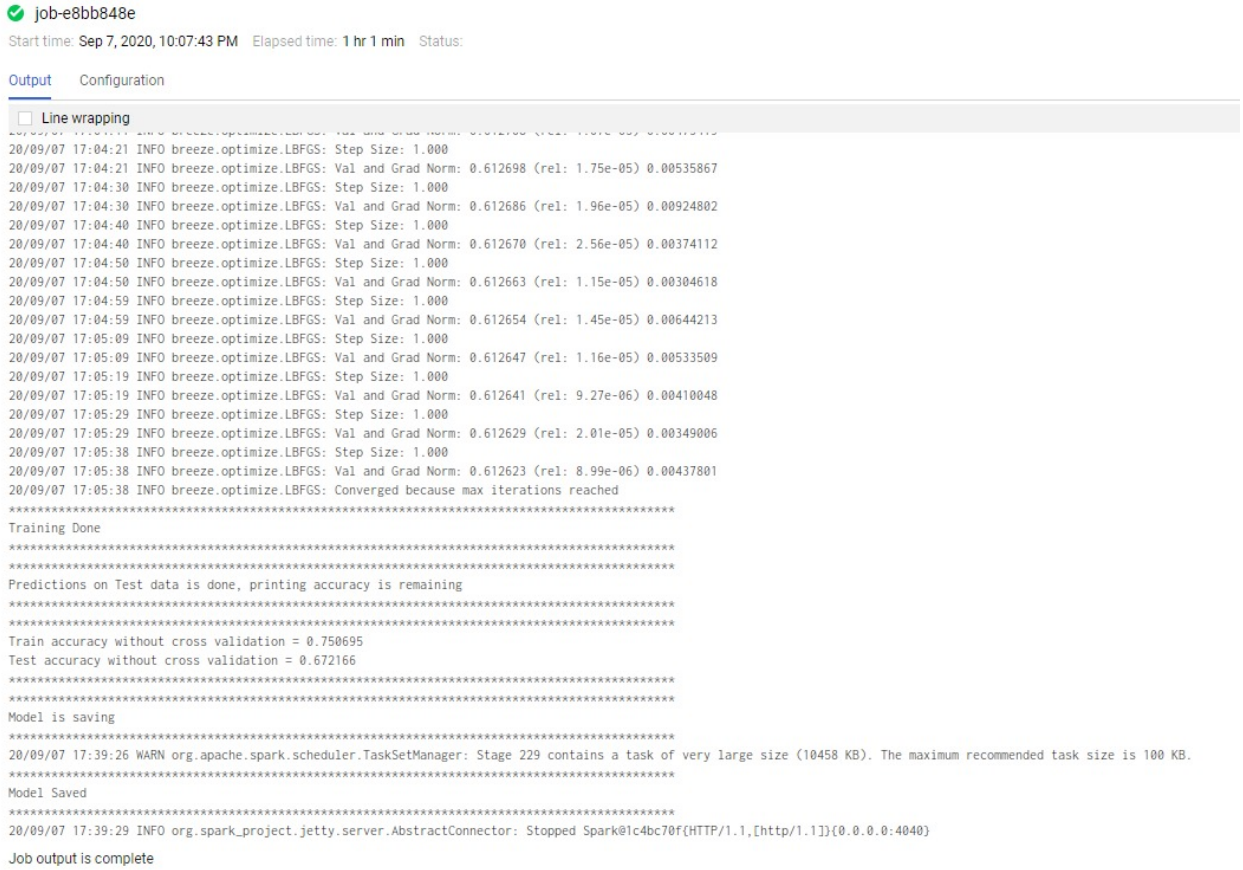


Figure 11: **Best Model (67.2% Accuracy)** - Logistic Regression with HashingTF and l2-Normalization

Train and Test Accuracy: The best accuracy given by the logistic regression built on the custom tfidf pyspark function and labels are modified from 0 to 4 instead of 1 to 5. Gave around **67.2%**

Observation:

- Logistic Regression performed the best as compared to the other two models.
- We used the same custom tfidf with l2-norm as was used in Naive Bayes Classifier.
- Hashing TF along with Stop Words were used for encoding the words.

5.3 Random Forest:

Parameters:

- numTrees=8,
- maxDepth=5,
- maxBins=32

Figure 12: Random Forest Model

Train and Test Accuracy: close to 45% on both train and test.

Observation:

- We tried RF with numTrees=64, but it took long to converge and hence we could not achieve the end result.
- So finally we reported accuracy obtained by the 8 trees model.

6 Real-time Computation

We use Kafka to build real-time data pipeline and integrate it with Pub/Sub to exchange messages between Kafka and Pub/Sub.

6.1 Streaming:

We launch and configure the Kafka VM Instance, create Pub/Sub Topic and Subscription. We use Cloud Shell to publish the test data in finite intervals as in Figure 13. This test data is then feed into the dataproc cluster which uses the pre-trained model to make the predictions Figure 14. We have also included a video along with the code Drive Link.

```
from google.cloud import storage
from google.cloud import pubsub_v1
import time

publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path('big-data-lab-266810', 'to-kafka')
client = storage.Client()
bucket = client.get_bucket('wcs_word')
blob = bucket.get_blob('yelp_test.csv')
x = blob.download_as_string()
x = x.decode("utf-8-sig")
x = x.split("\n")
futures = dict()

def get_callback(f, data):
    def callback(f):
        try:
            print(f.results())
        except:
            print("Please handle {} for {}.".format(f.exception(), data))
    return callback

for i in range(len(x)):
    data = x[i]
    if(len(data) < 10):
        continue
    futures.update({data: None})
    future = publisher.publish(topic_path, b'hi', val = data)
    futures[data] = future
    print(x[i])
    future.add_done_callback(get_callback(future, data))
    time.sleep(10)

print("Published Message with Error Handler")
```

Figure 13: Code to Publish Test Data from Cloud Shell

```
kafka_topic = 'from-pubsub'
zk = '10.182.0.2:2181'
app_name = "from-pubsub"
sc = SparkContext(appName="KafkaPubsub")
ssc = StreamingContext(sc, 0.1)
kafkaStream = KafkaUtils.createStream(ssc, zk, app_name, {kafka_topic: 1})
pipelineFit = PipelineModel.load('gs://wcs_word/NB_pipeline')
print("1")
model = NaiveBayesModel.load('gs://wcs_word/NB_FullTrainedModel')
print("2")
spark = SparkSession(sc)

init_time = None
count = 0

def row_generate(r):
    return Row(star = float(r[0]) -1, useful = float(r[1]), funny = float(r[2]), cool = float(r[3]), text = str(r[4]))

def lambda_func(r):
    return r[25:-2].split(',')

def func(ks):
    try:
        global init_time, count
        dat = ks.map(lambda r:r[1])
        dat1 = dat.map(lambda_func)
        # print("Start")
        for r in dat1.collect():
            count += 1
        # print("End")
        dat2 = dat1.map(row_generate)
        dat3 = spark.createDataFrame(dat2)
        dat3.show()
        dat_transform = pipelineFit.transform(dat3)
        pred = model.transform(dat_transform)
        pred.select("Prediction").show()
        print("recieved rdd")
        if(init_time == None):
            init_time = time.time()
        elif(time.time() - init_time > 9.9):
            time_elapsed = time.time() - init_time
            print("Messages Recieved: ", count)
            print("Time Elapsed: ", time_elapsed)
            print("Latency: ", count/time_elapsed)
            count = 0
            init_time = time.time()
    except:
        pass
```

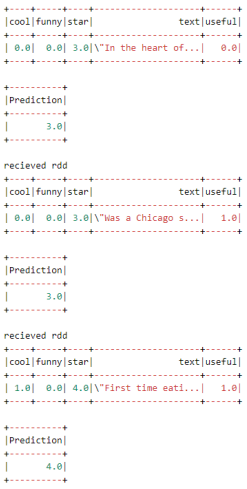
Figure 14: Prediction on test data points

6.2 Latency Calculation:

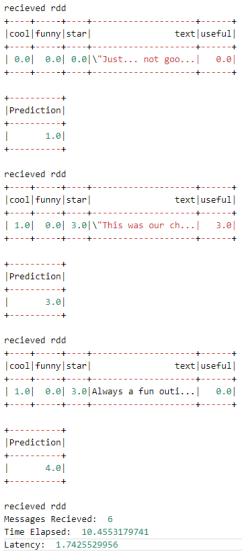
Latency means delay. For the project purposes, we define latency as the time taken for a message to be processed by the Kafka. This may involve bottlenecks such as:

- Bandwidth Limitation to transfer messages from Pub/Sub.
- Prediction Time.
- Waiting Time for data points to be mapped to a worker.
- Batch Duration for the streaming context.

We change the publish time interval from 10 to 0.5. This helps to decrease the delay caused automatically due to delay in sending messages. We cannot set it very low because in that case, Kafka will receive all the data points at once and will batch compute we don't want that to happen. We also set the Batch Duration



(a) Part - 1



(b) Part - 2

Figure 16: Latency with records

as 0.1. This is again to ensure that Kafka doesn't Batch Compute. We set a window of 10 seconds and count the number of data points processed during this window. We found that latency is around **1.743** as shown in Figure 16 and Figure 15. This may seem higher than expected but we think this because of the large size of the data points (long text).

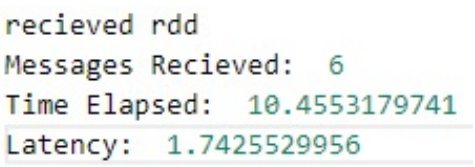


Figure 15: Calculated Latency