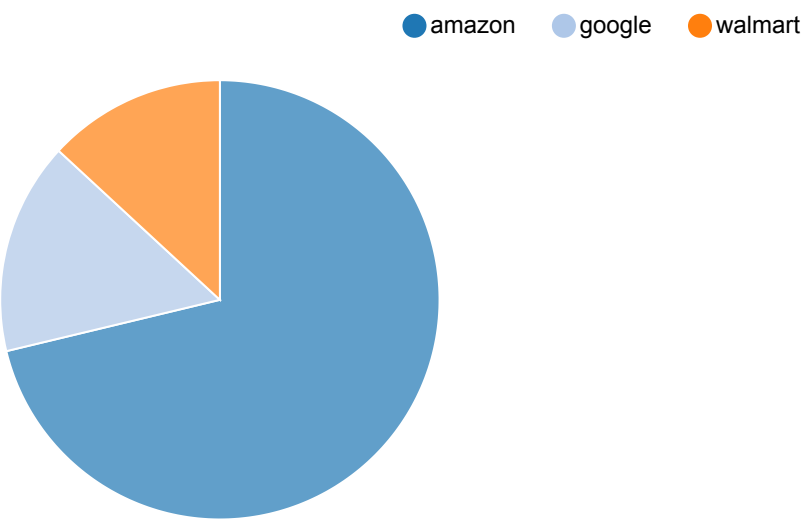
Tweets & Hashtags

Overall tweet count across usermentions

FINISHED

```
%jdbc(hive)
select tweet_usermention, count(distinct tweet_id) as tweet_count from tweet_data_table
    group by tweet_usermention;
```

settings



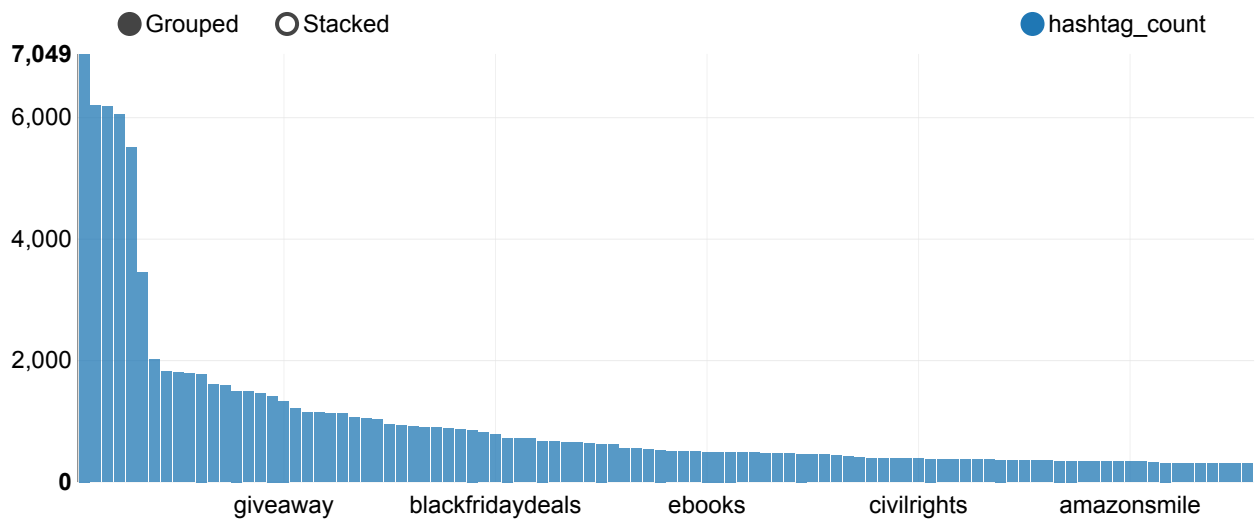
Took 33 sec. Last updated by anonymous at November 27 2018, 11:59:30 AM. (outdated)

Tweet count by hashtag

FINISHED

```
%jdbc(hive)
select tweet_hashtag, count(*) as hashtag_count from tweet_data_table group by
    tweet_hashtag order by hashtag_count desc limit 100;
```

settings



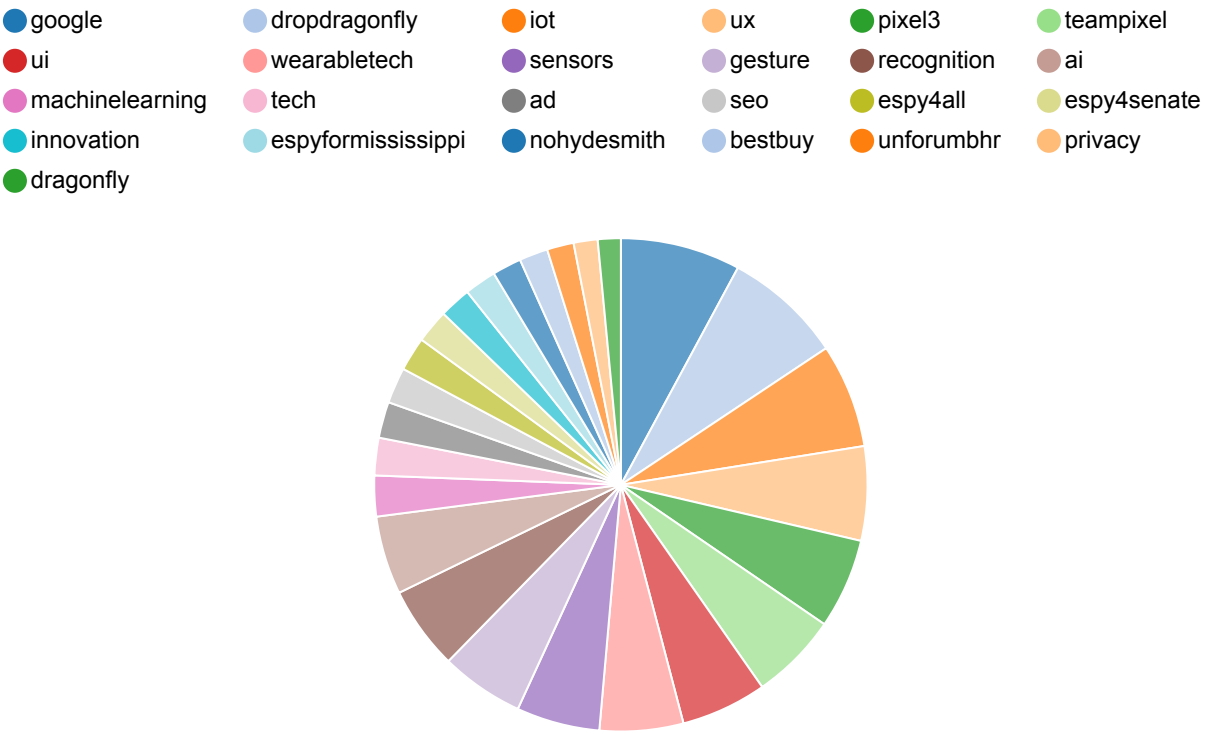
Took 44 sec. Last updated by anonymous at November 27 2018, 4:12:44 AM. (outdated)

Hashtag graph for Google

FINISHED

```
%jdbc(hive)
select tweet_hashtag, count(*) as hashtag_count from tweet_data_table where
    tweet_usermention = 'google' group by tweet_hashtag order by hashtag_count desc limit
    25 ;
```

settings



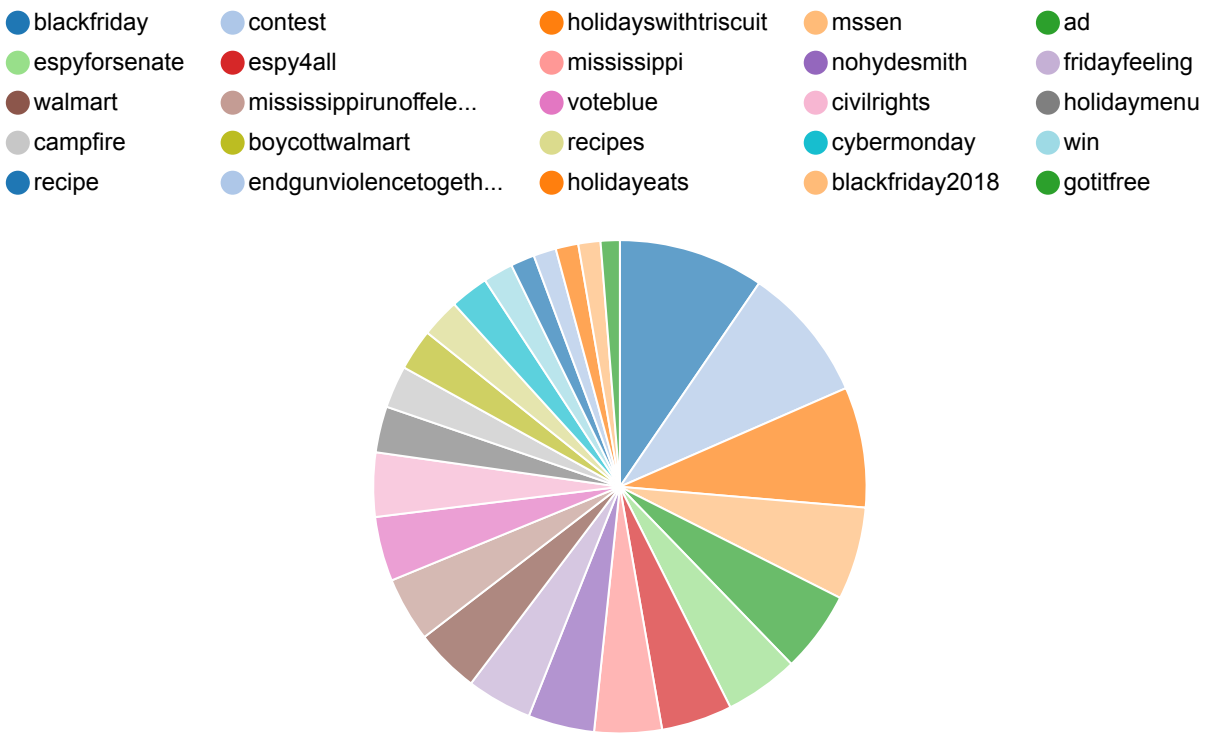
Took 42 sec. Last updated by anonymous at November 27 2018, 12:05:32 PM. (outdated)

Hashtag graph for Walmart

FINISHED

```
%jdbc(hive)
select tweet_hashtag, count(*) as hashtag_count from tweet_data_table where
    tweet_usermention = 'walmart' group by tweet_hashtag order by hashtag_count desc limit
    25 ;
```

settings



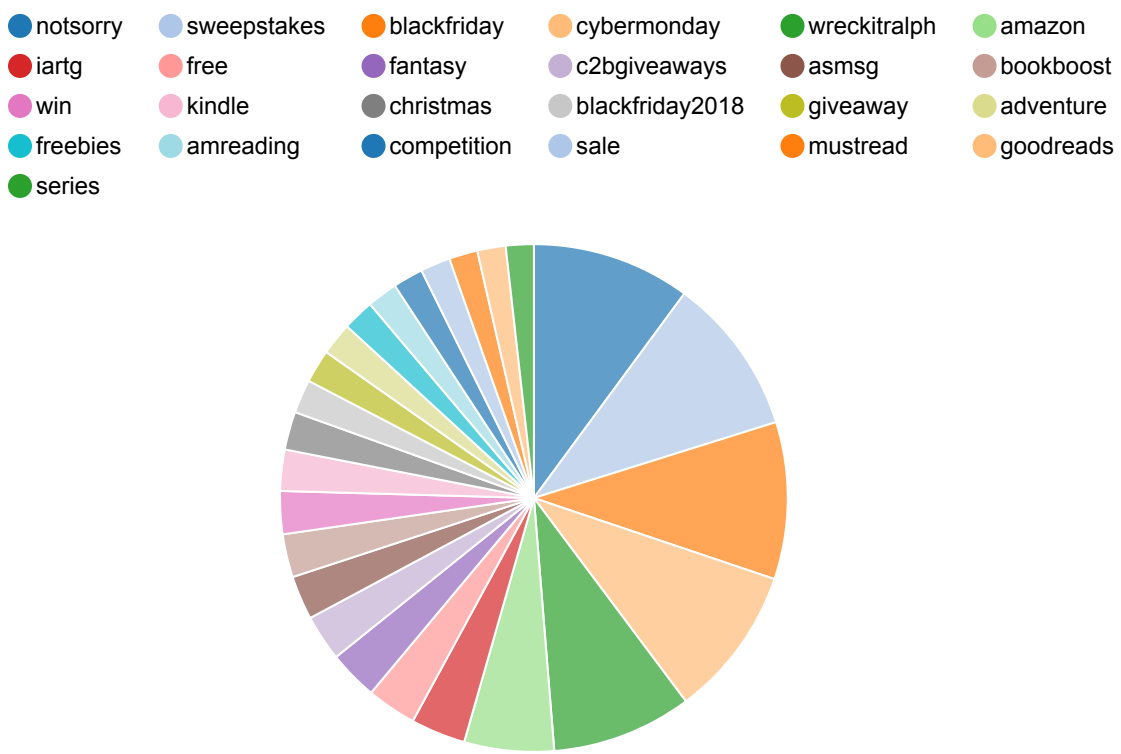
Took 44 sec. Last updated by anonymous at November 27 2018, 12:07:30 PM. (outdated)

Hashtag graph for Amazon

FINISHED

```
%jdbc(hive)
select tweet_hashtag, count(*) as hashtag_count from tweet_data_table where
    tweet_usermention = 'amazon' group by tweet_hashtag order by hashtag_count desc limit
    25 ;
```

settings

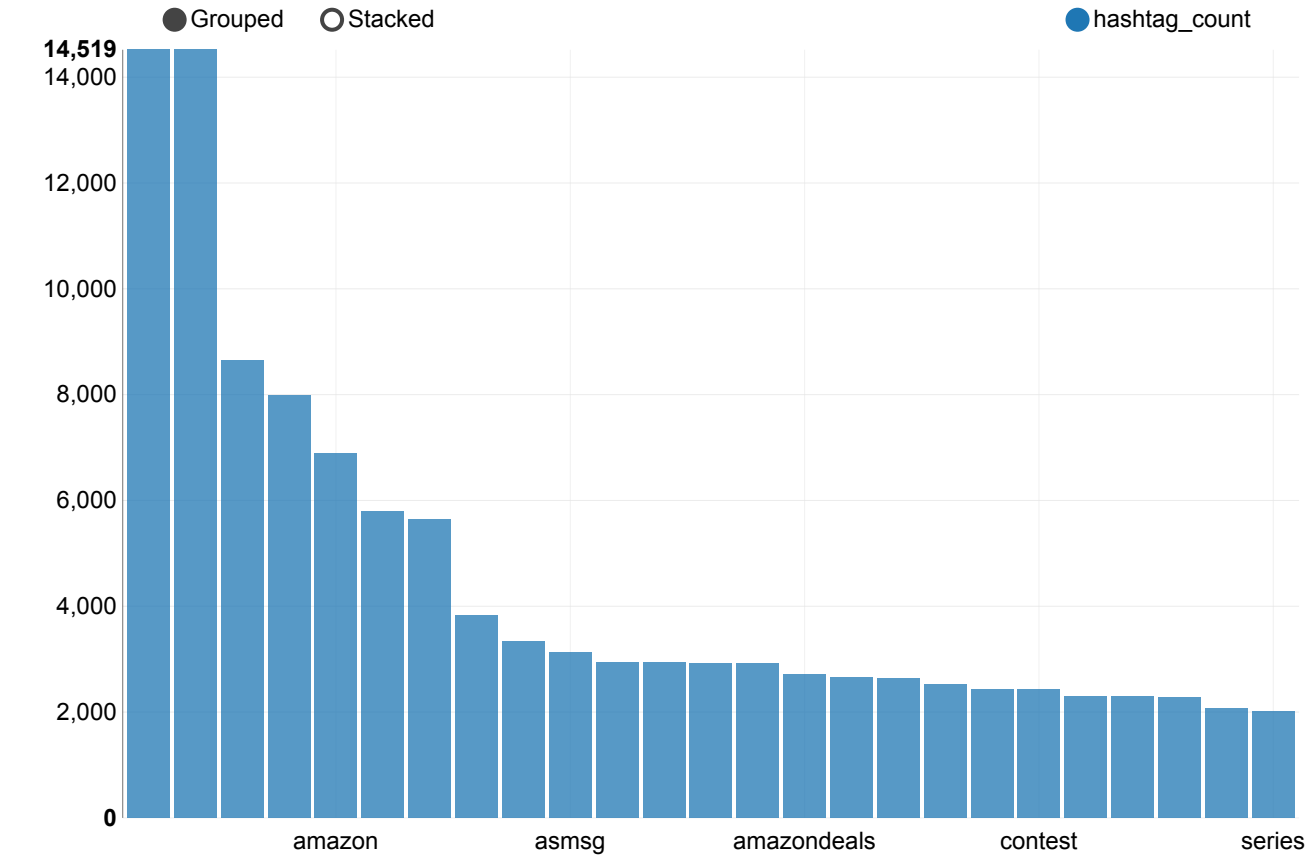


Took 43 sec. Last updated by anonymous at November 27 2018, 12:16:27 PM. (outdated)

```
%jdbc(hive)
select tweet_hashtag, count(*) as hashtag_count from tweet_data_table group by
    tweet_hashtag order by hashtag_count desc limit 25;
```

FINISHED

settings

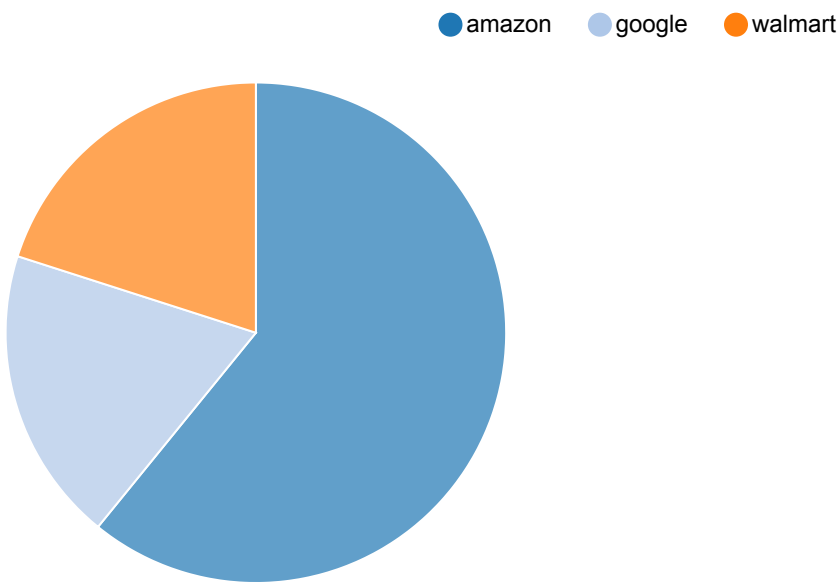


Took 43 sec. Last updated by anonymous at December 01 2018, 5:23:02 AM. (outdated)

```
%jdbc(hive)
select tweet_usermention, count(distinct tweet_id) as tweet_count from tweet_data_table
    group by tweet_usermention;
```

FINISHED

settings

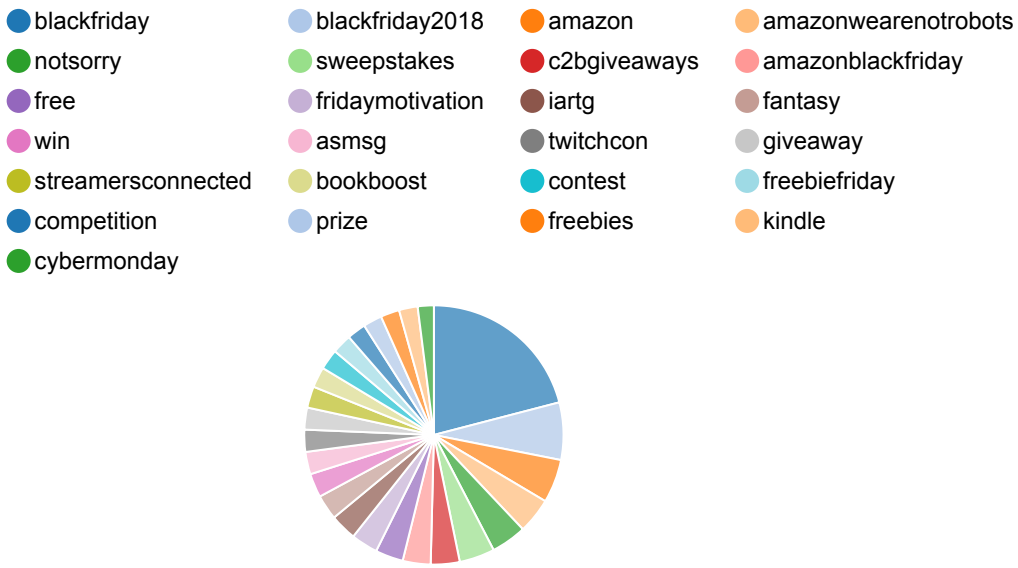


Took 21 sec. Last updated by anonymous at December 01 2018, 2:58:35 AM. (outdated)

```
%jdbc(hive)
select tweet_hashtag, count(*) as hashtag_count from tweet_data_table where
    tweet_usermention = 'amazon' and tweet_timestamp >= '2018-11-22' and tweet_timestamp <=
    '2018-11-24' group by tweet_hashtag order by hashtag_count desc limit 25;
```

FINISHED

settings



Took 43 sec. Last updated by anonymous at December 01 2018, 5:54:59 AM. (outdated)

```
%livy2.pyspark
import re

from pyspark import SparkContext
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression, NaiveBayes
from pyspark.ml.feature import Tokenizer, StopWordsRemover, CountVectorizer, IDF, HashingTI
from pyspark.sql import SparkSession
```

FINISHED

```
def train_filter_func(row):
    row_dict = row.asDict()
    val = row_dict['sentiment']
    if val in ['0', '4']:
        return True
    else:
        return False


def int_cast(row):
    row_dict = row.asDict()
    val = int(row_dict['sentiment'])
    if val == 0:
        row_dict['sentiment'] = 0
    else:
        row_dict['sentiment'] = 1
    return row_dict


def _clean_tweet_text(text):
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^A-Za-z \t])|(\w+:\/\/\S+)", " ", text).spli

def clean_train_tweet(row):
    '''
    Function to clean tweet text by removing links, special characters using regex .
    :param tweet_text: text of tweet recieved from database
    :return: cleaned text of tweet
    '''
    row['tweet_text'] = _clean_tweet_text(row['tweet_text'])
    return row


def clean_test_tweet(row):
    '''
    Function to clean tweet text by removing links, special characters using regex .
    :param tweet_text: text of tweet recieved from database
    :return: cleaned text of tweet
    '''
    row = row.asDict()
    row['tweet_text'] = _clean_tweet_text(row['tweet_text'])
    return row


################ Training data from Sentiment140  ######################################
train_df = spark.read.csv("hdfs:///user/maria_dev/sent_140.csv")
train_df = train_df.selectExpr("_c0 as sentiment", "_c5 as tweet_text").select('sentiment'
train_rdd = train_df.rdd
train_rdd = train_rdd.filter(train_filter_func).map(int_cast)
train_rdd = train_rdd.map(clean_train_tweet)
clean_train_df = spark.createDataFrame(train_rdd, schema="tweet_text: string, sentiment: i

############### Extracting orc datafile stored from HDFS ###############################
tweet_df = spark.read.orc('hdfs:///user/maria_dev/tweet_usermention')
tweet_df = tweet_df.filter(tweet_df.tweet_usermention=='amazon').filter(tweet_df.tweet_hash
#.filter("tweet_timestamp >= '2018-11-22' and tweet_timestamp <= '2018-11-25'")
tweet_df = tweet_df.select('tweet_text')
tweet_rdd = tweet_df.rdd
clean_tweet_rdd = tweet_rdd.map(clean_test_tweet)
clean_tweet_df = spark.createDataFrame(clean_tweet_rdd, "tweet_text: string")
clean_tweet_df = clean_tweet_df.dropDuplicates()

######## Feature Transformation ########################################################

############ Tokenize ##################################################################
```

```
tokenizer = Tokenizer(inputCol="tweet_text", outputCol="words")

############ Stop Words Remover ######################################################
remover = StopWordsRemover(inputCol="words", outputCol="filtered")

########### CountVectorizer ##########################################################
cv = CountVectorizer(inputCol="filtered", outputCol="cvfeatures", minDF=2.0)

hashtf = HashingTF(numFeatures=2 ** 16, inputCol="words", outputCol='tffeatures')

############Feature Extractors ########################################################

############ IDF #####################################################################
idf = IDF(inputCol='cvfeatures', outputCol="features",
          minDocFreq=5)  # minDocFreq: remove sparse terms # it down-weights
# columns which appear frequently in a corpus.
idf2 = IDF(inputCol='tffeatures', outputCol="features", minDocFreq=5)
############ Logistic Regression #####################################################
lr = LogisticRegression(labelCol="sentiment")
nb = NaiveBayes(labelCol="sentiment")

############ Pipelines ###############################################################
pipeline2 = Pipeline(stages=[tokenizer, remover, hashtf, idf2, lr])  # HashingTF + IDF + Lo
pipelineFit2 = pipeline2.fit(clean_train_df)
predictions2 = pipelineFit2.transform(clean_tweet_df)
pp = predictions2.select('prediction')
pp.write.csv("hdfs:///user/maria_dev/tweet_sentiment", mode="overwrite")
```

Previous livy session is expired, new livy session is created. Paragraphs that depend on this paragraph need to be re-executed!

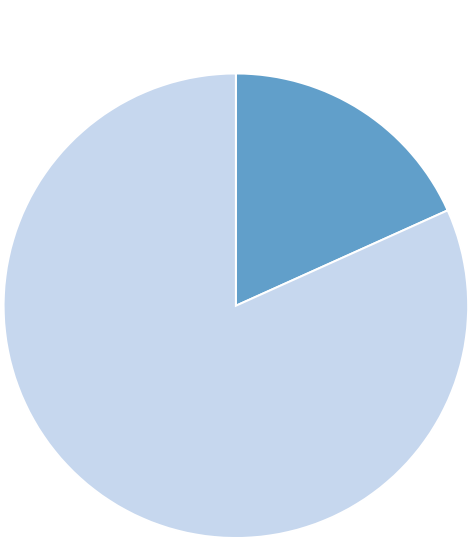
Spark Application Id: application_1543341205840_0089
Spark WebUI: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1543341205840_0089/ (http://sandbox-hdp.hortonworks.com:8088/proxy/application_1543341205840_0089/)

Took 2 min 41 sec. Last updated by anonymous at December 01 2018, 8:39:03 AM. (outdated)

Amazon Tweet Sentiment Analysis (CountVectorizer + IDF + Logistic Regression)

FINISHED

settings

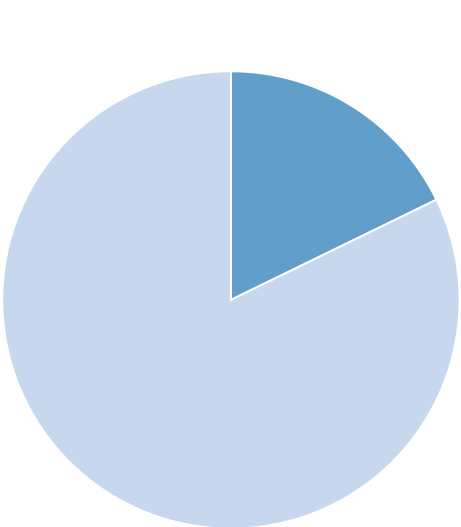


Took 43 sec. Last updated by anonymous at December 01 2018, 6:47:59 AM. (outdated)

Amazon Tweet Sentiment Analysis (HashingTF + IDF + Logistic Regression)

FINISHED

settings

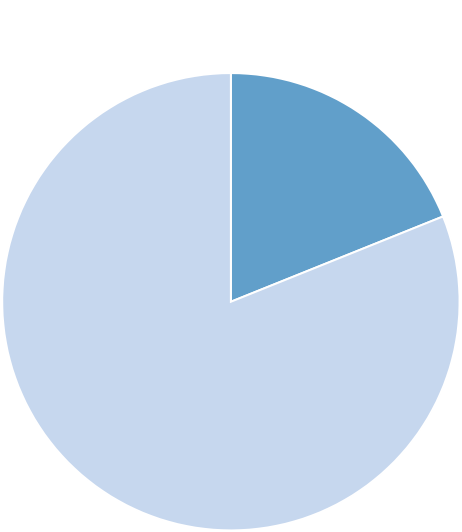


Took 23 sec. Last updated by anonymous at December 01 2018, 6:58:53 AM. (outdated)

Amazon Tweet Sentiment Analysis(HashingTF + IDF + Naive Bayes)

FINISHED

settings

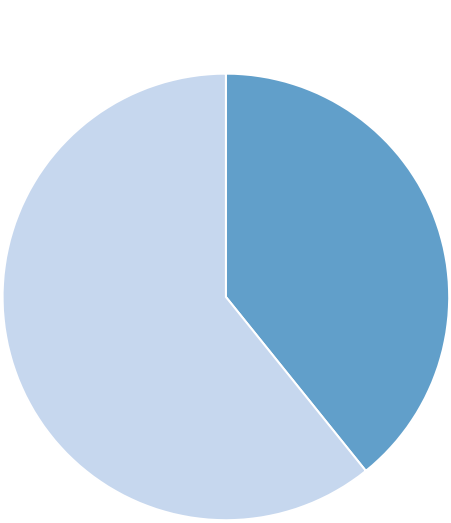


Took 51 sec. Last updated by anonymous at December 01 2018, 7:03:58 AM. (outdated)

Walmart Tweet Sentiment Analysis (HashingTF + IDF + Logistic Regression)

FINISHED

settings

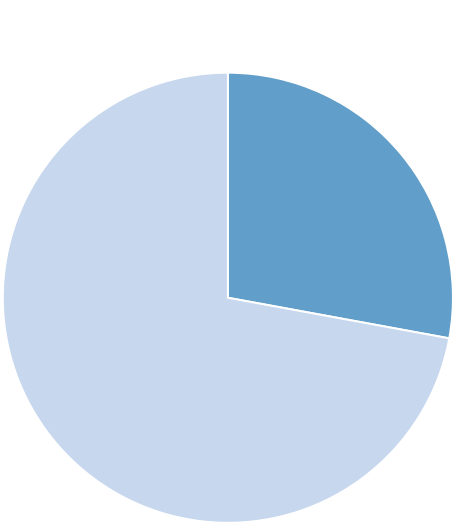


Took 20 sec. Last updated by anonymous at December 01 2018, 7:23:02 AM. (outdated)

Google Tweet Sentiment Analysis (HashingTF + IDF + Logistic Regression)

FINISHED

settings

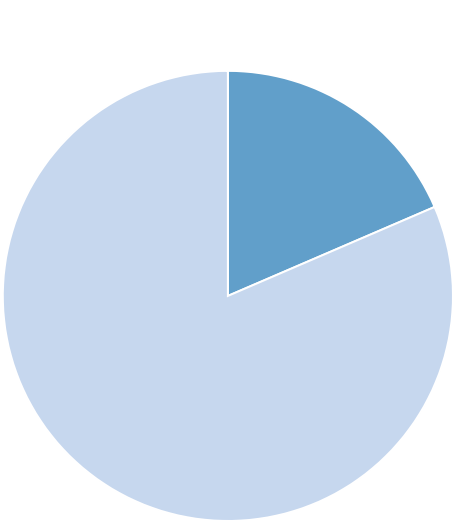


Took 20 sec. Last updated by anonymous at December 01 2018, 7:33:01 AM. (outdated)

Overall Amazon Tweet Sentiment Analysis (HashingTF + IDF + Logistic Regression)

FINISHED

settings

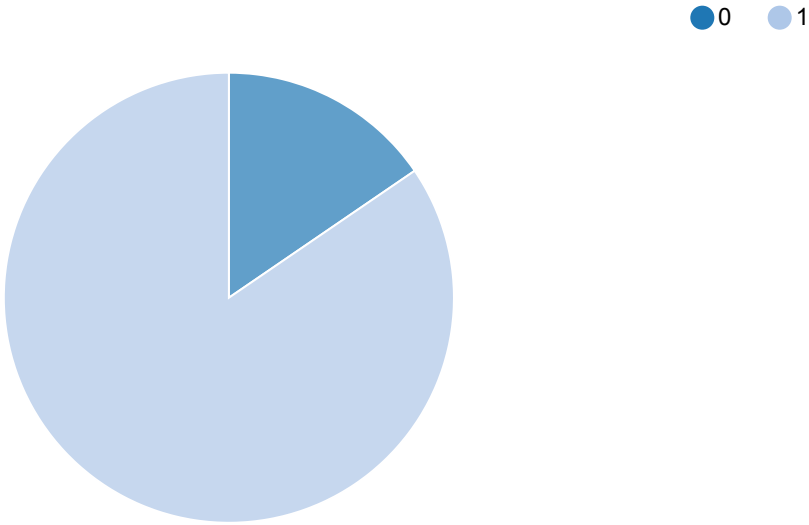


Took 22 sec. Last updated by anonymous at December 01 2018, 7:52:52 AM. (outdated)

Blackfriday overall till now for Amazon (HashingTF + IDF + Logistic Regression)

FINISHED

settings

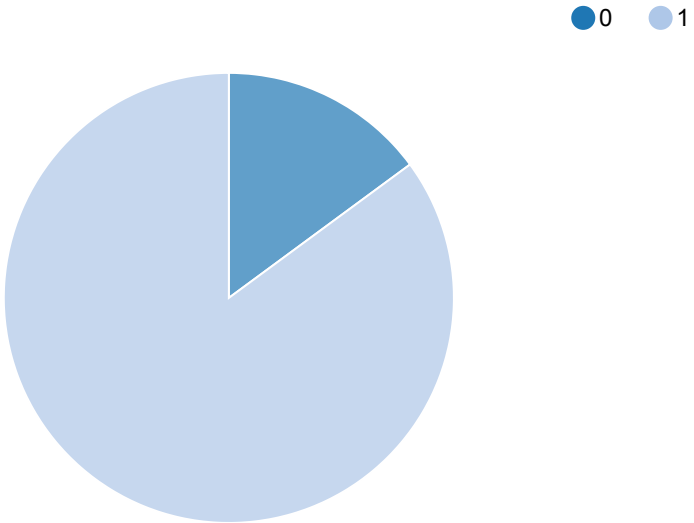


Took 33 sec. Last updated by anonymous at December 01 2018, 8:40:00 AM. (outdated)

Sentiment Analysis for Blackfriday for Amazon(HashingTF + IDF + Logistic Regression)

FINISHED

settings

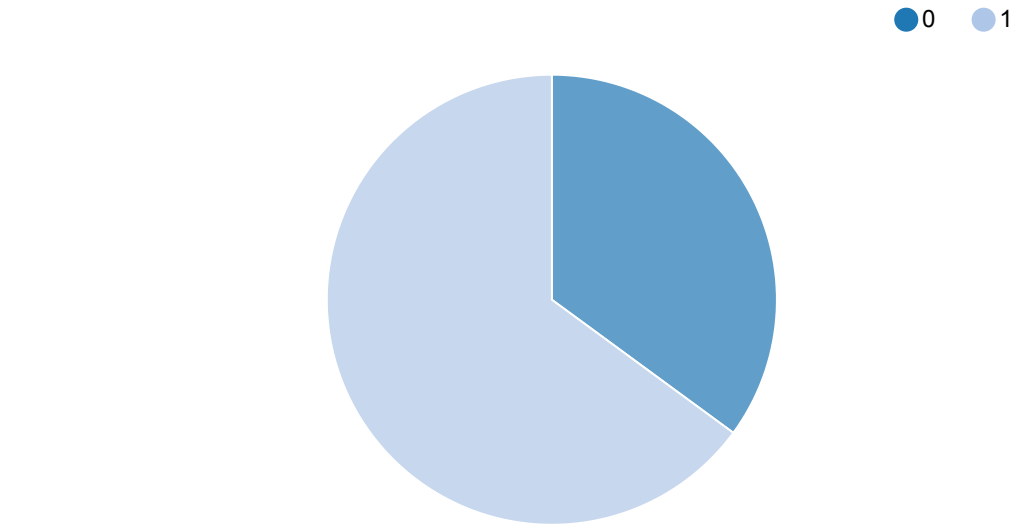


Took 20 sec. Last updated by anonymous at December 01 2018, 8:11:27 AM. (outdated)

Sentiment Analysis for Blackfriday for Walmart (HashingTF + IDF + Logistic Regression)

FINISHED

settings

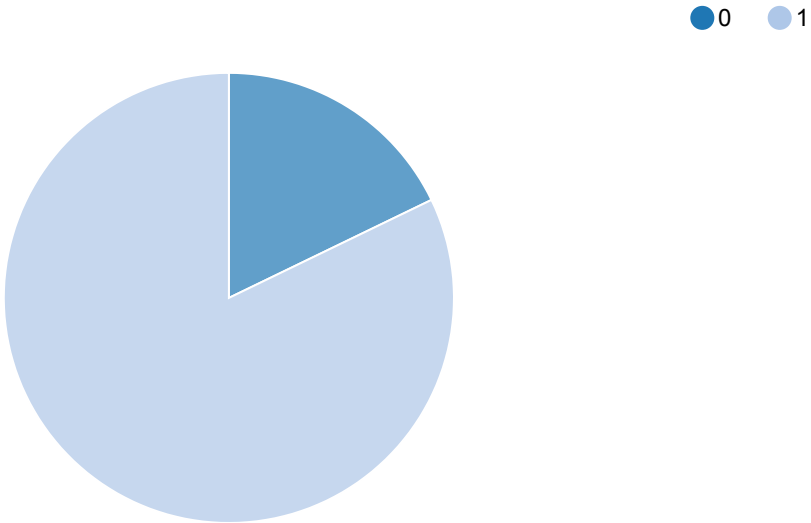


Took 23 sec. Last updated by anonymous at December 01 2018, 8:19:04 AM. (outdated)

Overall Blackfriday Sentiment Analysis across all (HashingTF + IDF + Logistic Regression)

FINISHED

settings



Took 22 sec. Last updated by anonymous at December 01 2018, 8:32:58 AM. (outdated)

READY