Cancer tweet sentiment analysis

Maggie and Ricardo
May 2nd, 2023

Big Data Project: Twitter Dashboard

Content

- Objectives
- Data
- Challenges
- Sentiment predictions
- Relevance of findings

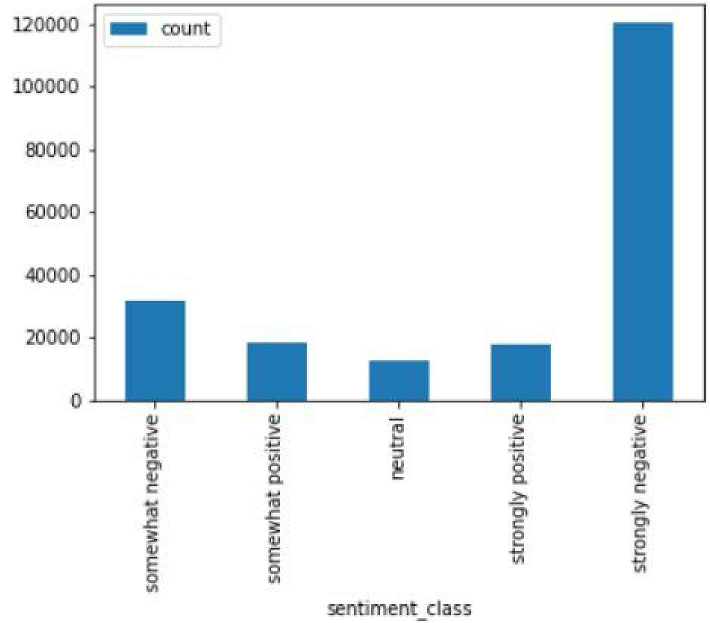
Objectives

- Apply Spark ML for a sentiment classification model
- Use Athena to create tables and Quicksight to build dashboards

Data

Columns: id, user_name, screen_name, tweet, followers, geo, coordinates, time

Rows: 201401 initial, 201067 after remove nulls



Processing:

Clean, tokenization, remove stopwords

count vectorization and TF-IDF

Problems and challenges

- Understanding Py-spark and how it works with AWS
- Free version is too slow takes a long time to predict models.
- Understanding datafiles as parquet and the advantages of each one.
- Slight difference in SQL queries with athena.
- Quicksight is less intuitive than Tableau and more limited in their visuals.

Sentiment Predictions

Logistic Regression: 1 Gram, Text Blob and binary classification.

Cmd 48

```
from pyspark.ml.feature import NGram, VectorAssembler, StopWordsRemover, HashingTF, IDF, Tokenizer, StringIndexer, CountVectorizer
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml import PipelineModel

# Use 70% cases for training, 30% cases for testing
train, test = df_for_model.randomSplit([0.7, 0.3], seed=42)

# Create transformers for the ML pipeline
tokenizer = Tokenizer(inputCol="tweet_text", outputCol="tokens")
stopword_remover = StopWordsRemover(inputCol="tokens", outputCol="filtered")
cv = CountVectorizer(vocabSize=2**16, inputCol="filtered", outputCol='cv')
idf = IDF(inputCol='cv', outputCol="1gram_idf", minDocFreq=5) #minDocFreq: remove sparse terms
assembler = VectorAssembler(inputCols=["1gram_idf"], outputCol="features")
# assembler convert several columns to one call features so it can be fed to the model
label_encoder= StringIndexer(inputCol = "sentiment_label", outputCol = "label")
# we always need a column call features and one call label, if not you need to go inside the model and change the default names
lr = LogisticRegression(maxIter=100)
lr_pipeline = Pipeline(stages=[tokenizer, stopword_remover, cv, idf, assembler,label_encoder,lr])

lr_pipeline_model = lr_pipeline.fit(train)
lr_predictions = lr_pipeline_model.transform(test)

lr_evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")
lr_accuracy = lr_predictions.filter(lr_predictions.label == lr_predictions.prediction).count() / float(test.count())
lr_roc_auc = lr_evaluator.evaluate(predictions)

print("Accuracy Score: {0:.4f}".format(accuracy))
print("ROC-AUC: {0:.4f}".format(roc_auc))


lr_pipeline_model.save('dbfs:/mnt/s3_bucket/Final_models/LR')
```

Accuracy: 0.86
ROC-AUC: 0.82

Logistic Regression: 1 & 2 Gram, Vader and multiclass classification.

```
# LogisticRegression Model

from pyspark.ml.feature import NGram, VectorAssembler, StopWordsRemover, HashingTF, IDF, Tokenizer, StringIndexer, NGram, ChiSqSelector, VectorAssembler
from pyspark.ml import Pipeline

# label encoder
from pyspark.ml.feature import StringIndexer

# label
label_encoder= StringIndexer(inputCol = "sentiment_class", outputCol = "label")

# Create transformers for the ML pipeline
tokenizer = Tokenizer(inputCol="tweet", outputCol="tokens")
stopword_remover = StopWordsRemover(inputCol="tokens", outputCol="filtered")
cv = CountVectorizer(vocabSize=2**16, inputCol="filtered", outputCol='cv')
idf = IDF(inputCol='cv', outputCol="1gram_idf", minDocFreq=5) #minDocFreq: remove sparse terms
ngram = NGram(n=2, inputCol="filtered", outputCol="2gram")
ngram_hashingtf = HashingTF(inputCol="2gram", outputCol="2gram_tf", numFeatures=20000)
ngram_idf = IDF(inputCol='2gram_tf', outputCol="2gram_idf", minDocFreq=5)

# assemble multiple input columns into a vector column, and then perform feature selection on the resulting vector column using the chi-squared test
# Assemble all text features
assembler = VectorAssembler(inputCols=["1gram_idf", "2gram_tf"], outputCol="rawFeatures")

# Chi-square variable selection
selector = ChiSqSelector(numTopFeatures=2**14,featuresCol='rawFeatures', outputCol="features")

# Regression model estimator
lr = LogisticRegression(maxIter=100)

# Build the pipeline
pipeline = Pipeline(stages=[label_encoder, tokenizer, stopword_remover, cv, idf, ngram, ngram_hashingtf, ngram_idf, assembler, selector, lr])

# Pipeline model fitting
pipeline_model = pipeline.fit(trainDF)
predictions = pipeline_model.transform(testDF)

evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
accuracy = predictions.filter(predictions.label == predictions.prediction).count() / float(testDF.count())
roc_auc = evaluator.evaluate(predictions)

print("Accuracy Score: {0:.4f}".format(accuracy))
print("ROC-AUC: {0:.4f}".format(roc_auc))
```

Accuracy: 0.8724
ROC-AUC: 0.8728

Raw data

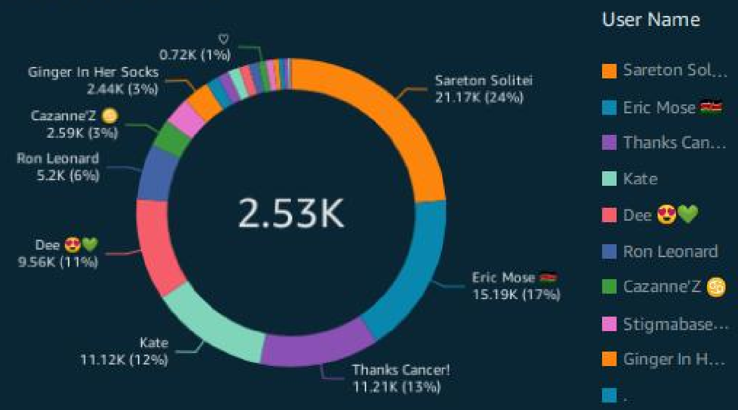
Tweet Count per hour GMT+0

WordCloud before cleaning data

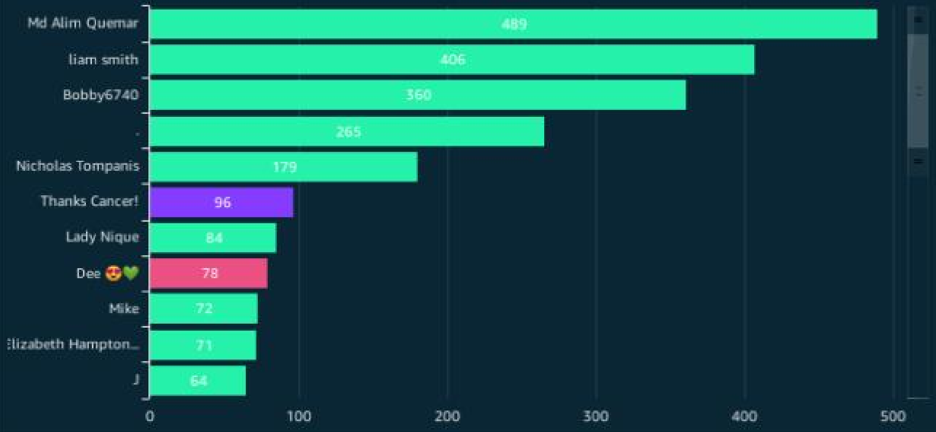
SHOWING TOP 100 IN TWEET_TEXT

Average followets by user

SHOWING TOP 20 IN USER_NAME



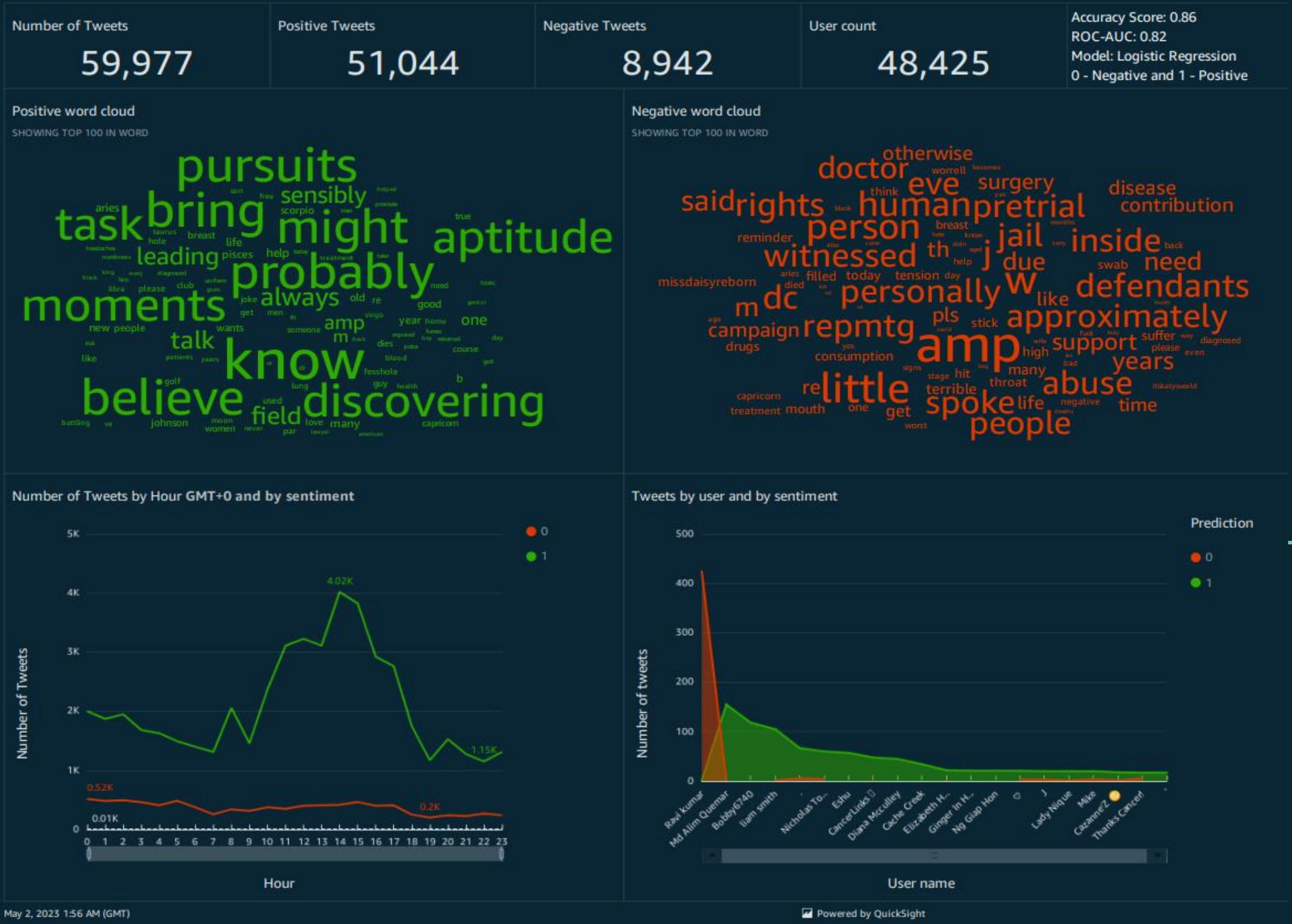
Tweets created from Nov 22 to Nov 24 2022 by user



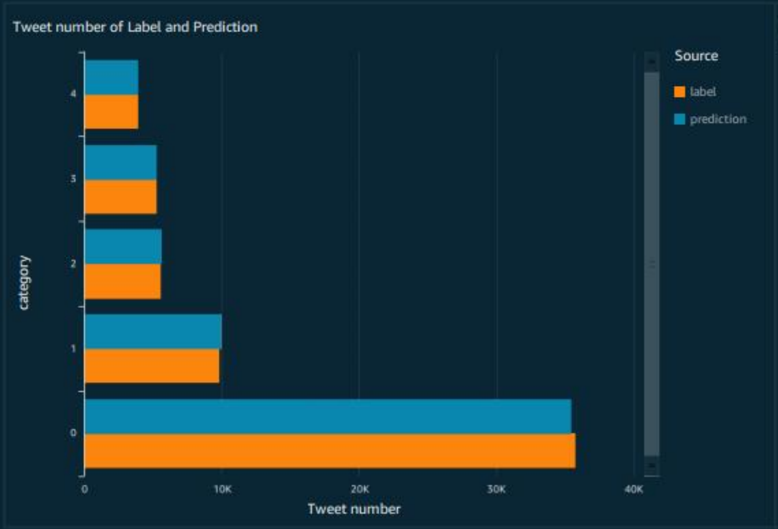
May 2, 2023 1:52 AM (GMT)

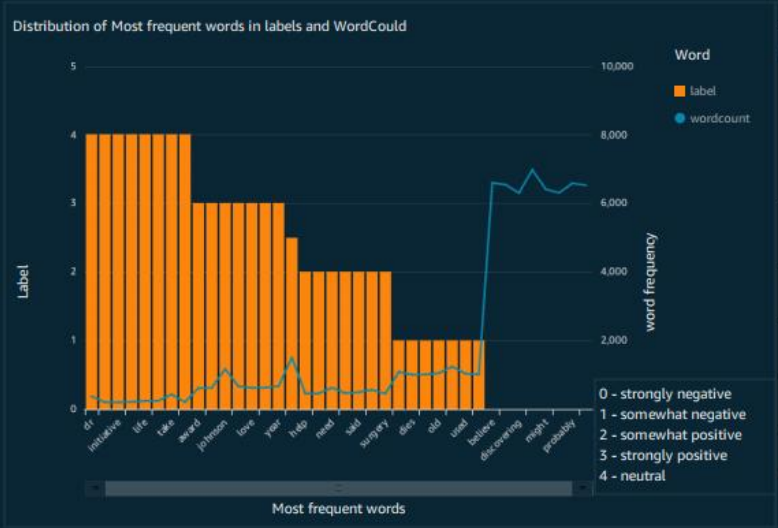
Powered by QuickSight

Prediction with Text Blob & 1 gram



Prediction with VADER 1 & 2 grams.





Conclusion

- The sentiment labels from TextBlob and VADER are very different, and neither of them provides a fully fitted sentiment analysis.
- Logistic Regression was the best model in both cases and performed better than Random Forest.
- Experiment with a dataset that has the manually processed labels
- Takes emoticons into consideration
- Further analysis of N-grams and difference in multiclass and binary labels.