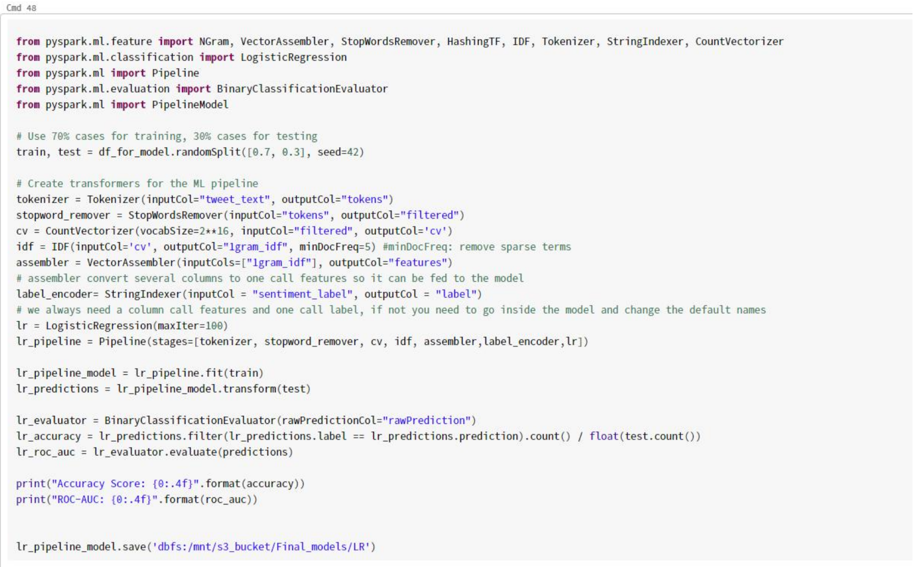
Sentiment Predictions

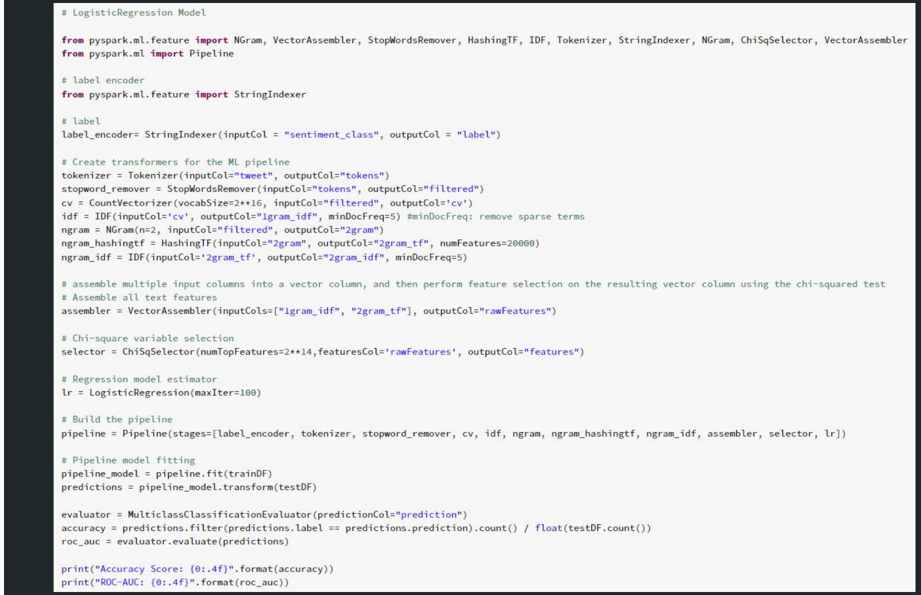
Logistic Regression: 1 Gram, Text Blob and binary classification.



```
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

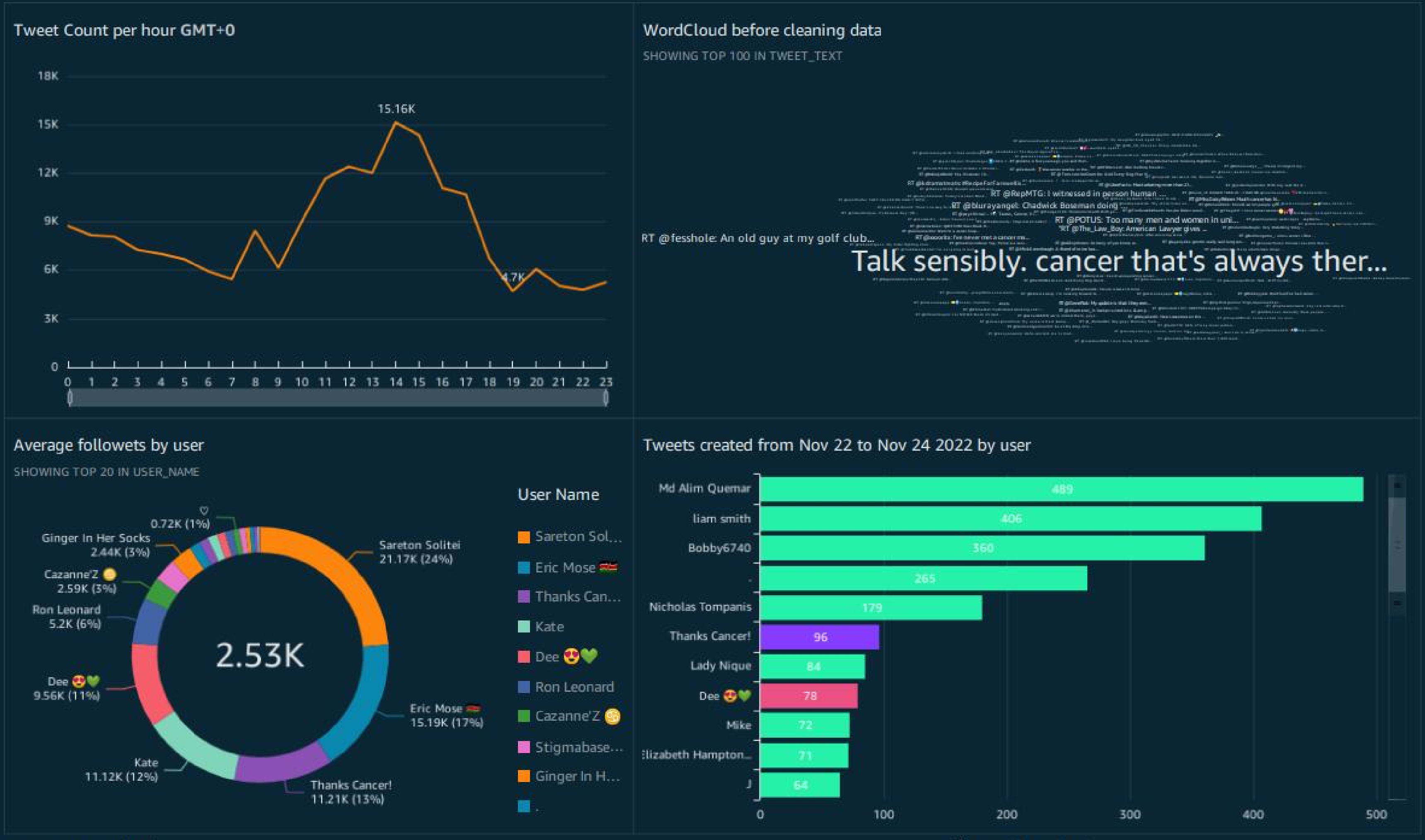
Logistic Regression: 1 & 2 Gram, Vader and multiclass classification.



```
# 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

RT @fesshole: An old guy at my golf club...
Talk sensibly. cancer that's always ther...
RT @RepMTG: I witnessed in person human ...
RT @blurayangel: Chadwick Boseman doing ...
RT @POTUS: Too many men and women in uni...
RT @The_Law_Boy: American Lawyer gives ...

Average followets by user

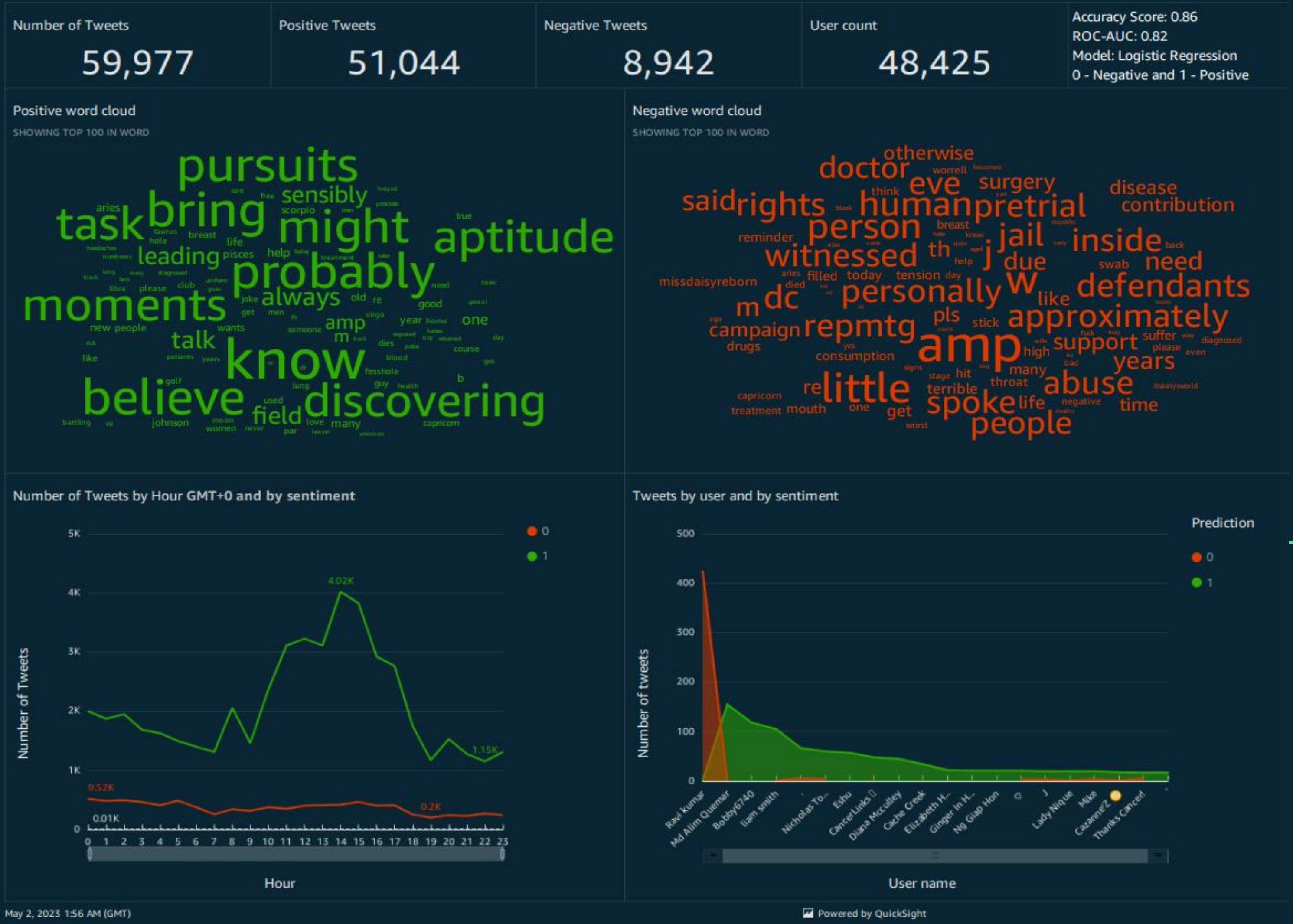
SHOWING TOP 20 IN USER_NAME

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



Number of Tweets	Positive Tweets	Negative Tweets	User count
59,977	51,044	8,942	48,425

Accuracy Score: 0.86
ROC-AUC: 0.82
Model: Logistic Regression
0 - Negative and 1 - Positive

Positive word cloud
SHOWING TOP 100 IN WORD

Negative word cloud
SHOWING TOP 100 IN WORD

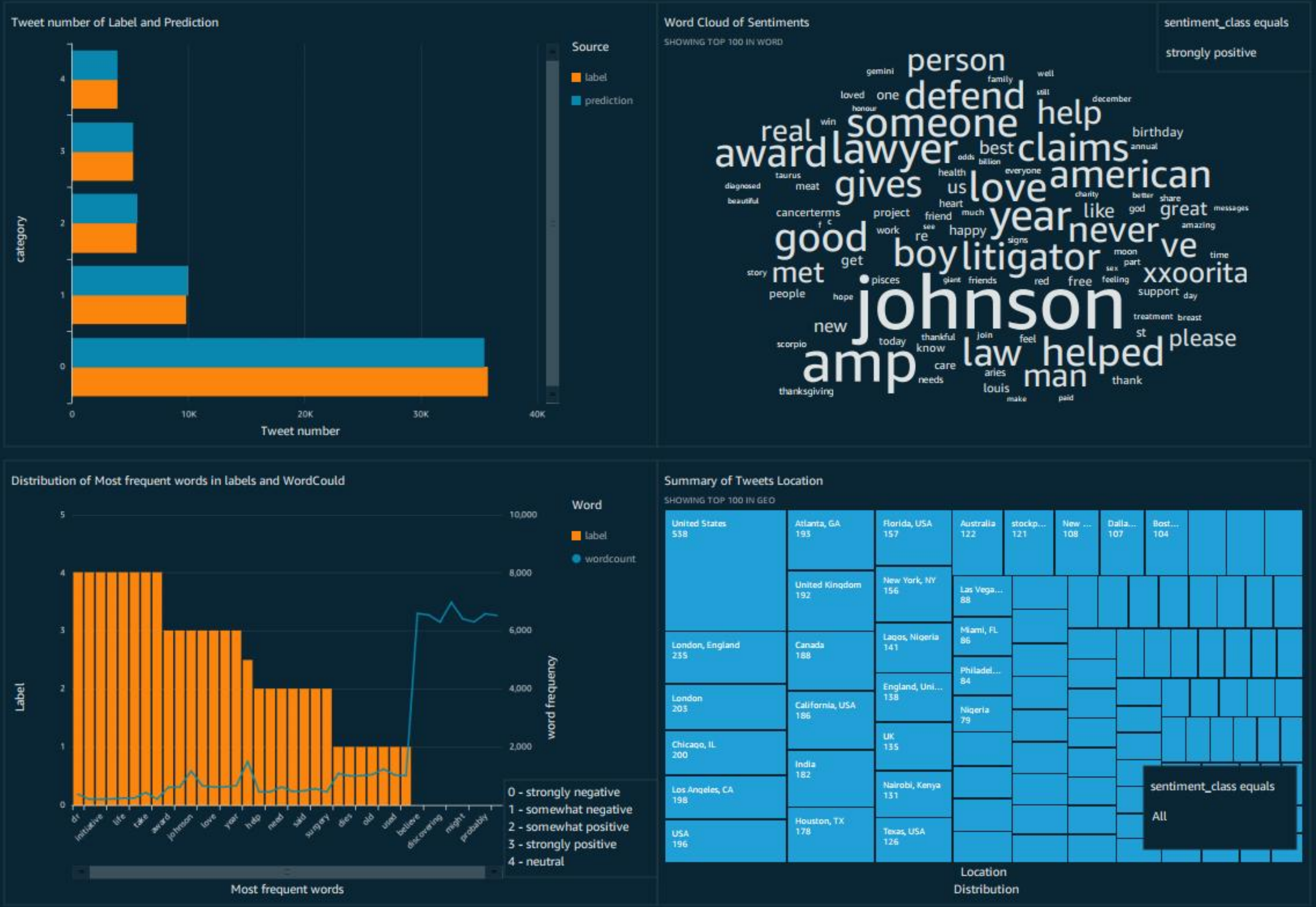
Number of Tweets by Hour GMT+0 and by sentiment

Tweets by user and by sentiment

May 2, 2023 1:56 AM (GMT)

Powered by QuickSight

Prediction with VADER 1 & 2 grams.



Tweet number of Label and Prediction

Word Cloud of Sentiments

SHOWING TOP 100 IN WORD

sentiment_class equals

strongly positive

Distribution of Most frequent words in labels and WordCould

0 - strongly negative
1 - somewhat negative
2 - somewhat positive
3 - strongly positive
4 - neutral

Summary of Tweets Location

SHOWING TOP 100 IN GEO

sentiment_class equals

All