Big Data
Twitter Sentiment Analysis

Maria Opekhtina

DS-B17

May, 2023

Agenda

- Introduction and Objective
- Data
- Machine Learning in Databricks
- SQL queries in Athena
- QuickSight Visualizations
- Challenges
- Next Steps

Introduction

- About FIFA World Cup
- International football competition of the men’s national teams
- Held every 4 years since 1930
- As of the 2022, 22 final tournaments have been held and a total of 80 national teams have competed
- WC 2022 took place in Qatar from 20 November to 18 December 2022
- Engagement with 2022 World Cup was estimated to be around 5 billion with close to 1.5 billion people watching the final match

- Project Objective

- Analyze Twitter sentiment during World Cup 2022 using Big Data tools and get a deeper understanding of how users engage with an event of this scale

Dataset

- Original size: 22 million rows
- Data used in this project: 998000 rows
- VADER(Valence Aware Dictionary for Sentiment Reasoning) NLTK module was used to extract sentiment and create labels

Cleaning

```
# Removing data that is purely retweets
data_0 = data.filter(col('text').like('RT%')==False)
```

```
# Limiting the dataframe to the first 1 million rows
data_1 = data_0.limit(1000000)
```

Loading Data

```
# Mounting WCD bucket with the data
mount_s3_bucket(ACCESS_KEY, SECRET_ACCESS_KEY, 'weclouddata/twitter/WorldCup/', 'project')
```

```
file = '/mnt/project/*/*/*/*/*'

data = (spark.read
    .option('header', 'false')
    .option('delimiter', '\t')
    .schema(wcSchema)
    .csv(file)
)
```

Creating Label

```
def get_sentiment_udf(text_series: pd.Series) -> pd.Series:
    analyzer = SentimentIntensityAnalyzer()
    sentiments = []
    for text in text_series:
        sentiment = analyzer.polarity_scores(text)['compound']
        if sentiment > 0:
            sentiments.append('positive')
        elif sentiment < 0:
            sentiments.append('negative')
        else:
            sentiments.append('neutral')
    return pd.Series(sentiments)
```

```
# using pandas_udf provided the speediest processing time
get_sentiment_pandas_udf = pandas_udf(get_sentiment_udf, returnType=StringType())

# creating columnt 'sentiment' using user-defined function 'get sentiment'
df = df.withColumn('sentiment', get_sentiment_pandas_udf(col('text')))
```

Final Table

	id	user_name	user_screen_name	text	followers_count	location	created	sentiment
1	1601642512845705216	ayub abdulahi max'ud	ayub_ud	bayc world cup nft @Jftblockchain @Fadedbaoge @ms_hennessey1 @3taizi666 @miniwhalecrypto @sovereigntom @VegabondETH... https://t.co/K0l99vuAEH	3	None	2022-12-10T18:18:26.000+0000	neutral
2	1601642513080922112	Watch Plug Wobs ⚡ 🍷	_varg76	😂😂😂💀 Gbafest	254	Nigeria	2022-12-10T18:18:26.000+0000	neutral
3	1601642513366155264	🙄	thoughtsofdebs	Yayyyyyy if Nigeria doesn't make it to the next WC something is seriously wrong 😭	322	London	2022-12-10T18:18:26.000+0000	negative
4	1601642511671230464	Better Odds Prediction (BOP)	Manaji111	I feel the pain for him, it's his last World cup appearance, he is a great man, at the age of 37 he can still displ... https://t.co/c4Z6FZpqnl	111	None	2022-12-10T18:18:26.000+0000	positive
5	1601642513957552128	Kulani M	kulani_kulls	Messi has more World Cup goals than him by the way #FIFAWorldCup	21011	Ebony Park, Gauteng	2022-12-10T18:18:27.000+0000	neutral
6	1601642513634275329	Oye-Asif	Asiif_tweeets	If israel played at World Cup... #FIFAWorldCup https://t.co/rO6iUuGUU7	5057	Hum Gilgit Baltistan Ka Hai.	2022-12-10T18:18:26.000+0000	positive

Machine Learning

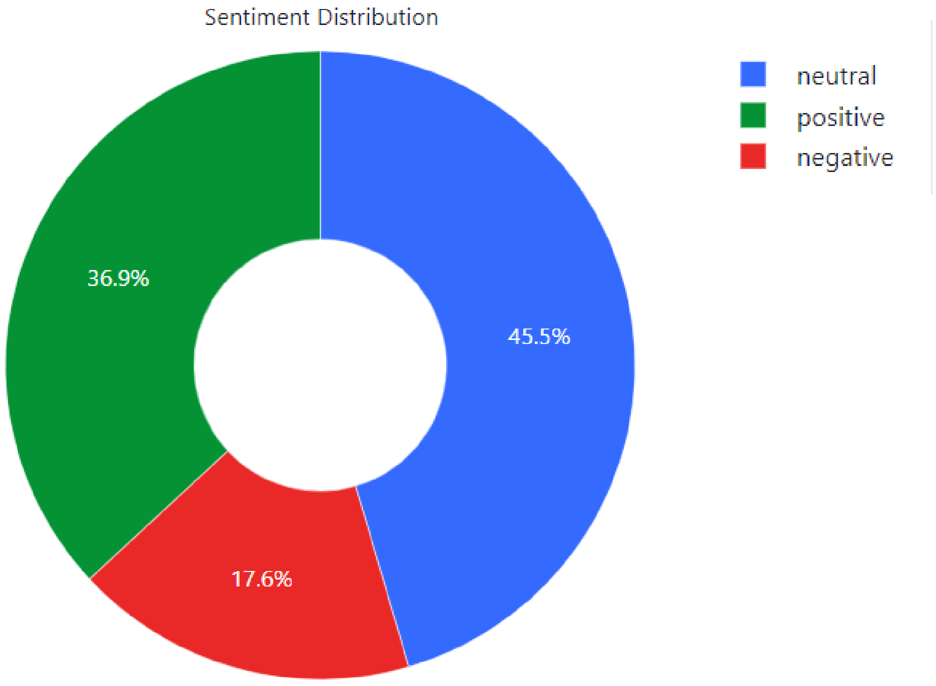
Data Preparation

```
# Dropping all columns but text and sentiment
tweets = df.select(col('text'), col('sentiment'))
display(tweets)
```

```
# Removing extra spaces, symbols, lowering text and trimming empty spaces
tweets_clean = (tweets
              .withColumn('text', regexp_replace(col('text'), r'http\S+', ''))
              .withColumn('text', regexp_replace(col('text'), r'[^a-zA-Z\s]', ' '))
              .withColumn('text', lower(col('text')))
              .withColumn('text', trim(col('text'))))
display(tweets_clean)
```

```
# tokenizing data
tokenizer = Tokenizer(inputCol='text', outputCol='tokens')
train_tokenized = tokenizer.transform(train)

# removing stopwords
stopword_remover = StopWordsRemover(inputCol='tokens', outputCol='filtered')
train_stopword = stopword_remover.transform(train_tokenized)

cv = CountVectorizer(vocabSize=2**16, inputCol='filtered', outputCol='cv')
cv_model = cv.fit(train_stopword)
train_cv = cv_model.transform(train_stopword)

# using idf to get word importances
idf = IDF(inputCol='cv', outputCol='features', minDocFreq=5)
idf_model = idf.fit(train_cv)
train_idf = idf_model.transform(train_cv)

# using label encoder to index sentiment
label_encoder = StringIndexer(inputCol = 'sentiment', outputCol = 'label')
le_model = label_encoder.fit(train_idf)
train_final = le_model.transform(train_idf)

display(train_final)
```

Label Distribution



Machine Learning Models

Basic LR

```
lr = LogisticRegression(maxIter=100)

lr_model = lr.fit(train_final)

predictions = lr_model.transform(train_final)

display(predictions)
```

```
# Model evaluation - accuracy and f1
evaluator_acc = MulticlassClassificationEvaluator(predictionCol='prediction', metricName='accuracy')
evaluator_f1 = MulticlassClassificationEvaluator(predictionCol='prediction', metricName='f1')

accuracy = evaluator_acc.evaluate(predictions)
f1_score = evaluator_f1.evaluate(predictions)


print('Accuracy Score: {0:.4f}'.format(accuracy))
print('F1 Score: {0:.4f}'.format(f1_score))
```

Model	Accuracy	F1
Logistic Regression	0.9478	0.9476
Decision Tree	0.9476	0.4875
Naive Bayes	0.7551	0.7596

GridSearchCV on LR

Python

```
# Defining the classifiers and the parameter grids
lr = LogisticRegression(maxIter=100)

lr_param_grid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.1, 0.01]) \
    .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
    .build()

evaluator_f1 = MulticlassClassificationEvaluator(metricName='f1')

cv_lr = CrossValidator(estimator=lr, estimatorParamMaps=lr_param_grid, evaluator=evaluator_f1, numFolds=3)

# Fitting the grid search
cv_lr_model = cv_lr.fit(train_final)

# Grabbing the best parameters from the gridsearch
best_metric_index = max(range(len(cv_lr_model.avgMetrics)), key=cv_lr_model.avgMetrics.__getitem__)
best_params = cv_lr_model.getEstimatorParamMaps()[best_metric_index]
display(best_params)

# Best f1 score
best_score = max(cv_lr_model.avgMetrics)
print(best_score)
```

Best Score 0.8677602871154536

Machine Learning Final Model

Logistic Regression – test set using hyperparameters from GridSearch

Python

```
train, test = tweets_clean.randomSplit([0.8, 0.2], seed=42)

# Creating transformers for the ML pipeline
tokenizer = Tokenizer(inputCol='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)
assembler = VectorAssembler(inputCols=['1gram_idf'], outputCol='features')
label_encoder= StringIndexer(inputCol = 'sentiment', outputCol = 'label')
lr = LogisticRegression(maxIter=100, regParam=0.01, elasticNetParam=0.0)

pipeline = Pipeline(stages=[tokenizer, stopword_remover, cv, idf, assembler, label_encoder, lr])

pipeline_model = pipeline.fit(train)
predictions = pipeline_model.transform(test)

accuracy = predictions.filter(predictions.label == predictions.prediction).count() / float(test.count())
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", metricName="f1")
f1_score = evaluator.evaluate(predictions)

print('Accuracy Score: {0:.4f}'.format(accuracy))
print('F1 Score: {0:.4f}'.format(f1_score))
```

(51) Spark Jobs

train: pyspark.sql.dataframe.DataFrame = [text: string, sentiment: string]

test: pyspark.sql.dataframe.DataFrame = [text: string, sentiment: string]

```
Accuracy Score: 0.9229
F1 Score: 0.9224
```

Logistic Regression – full dataset using hyperparameters from GridSearch

```
# Creating transformers for the ML pipeline
tokenizer = Tokenizer(inputCol='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

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)

# Assembling 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')

label_encoder= StringIndexer(inputCol = 'sentiment', outputCol = 'label')
lr = LogisticRegression(maxIter=100, regParam=0.01, elasticNetParam=0.0)

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

pipeline_model = pipeline.fit(tweets_clean)
predictions_full = pipeline_model.transform(tweets_clean)

accuracy = predictions_full.filter(predictions_full.label == predictions_full.prediction).count() /
float(tweets_clean.count())
evaluator = MulticlassClassificationEvaluator(predictionCol='prediction', metricName='f1')
f1_score = evaluator.evaluate(predictions_full)

print('Accuracy Score: {0:.4f}'.format(accuracy))
print('F1 Score: {0:.4f}'.format(f1_score))
```

(51) Spark Jobs

```
Accuracy Score: 0.9401
F1 Score: 0.9398
```

Machine Learning Exporting Predictions

Cleaning

```
# Dropping unneeded columns and saving the dataframe to be exported into s3
str_pred = predictions.withColumn('filtered', col('filtered')\   # Cleaning up filtered column as csv does not take arrays
                        .cast('string'))\
                        .withColumn('filtered', regexp_replace(col('filtered'), r'[^a-zA-Z\s]', ' '))

prediction = str_pred.select('text', 'filtered', 'sentiment', 'label', 'prediction')
display(prediction)
```

Saving to S3

```
# Saving predictions on full set
s3_path = 'b17-masha/project/predictions_full/'

# Write the DataFrame to the mounted S3 bucket
(prediction_full.write
  .format('csv')
  .option('header', True)
  .option('delimiter', '\t')
  .mode('overwrite')
  .save(f'/mnt/storage/{s3_path}')
)
```

Dataset

filtered	sentiment	label	prediction
	neutral	0	0
lead feels dangerous far world cup leads complacency tension probably keeps everyone toes	negative	2	2
year old boy killed montpellier amid clashes france morocco fans following fif	negative	2	2
french rendez vous awaits class really shone bellingham exceptional talent	positive	1	1
walow nft alphonso davies qatar world cup listed auction weth collection	neutral	0	0
ba phalaze ba futhe ba chathe ll fine tomorrow	positive	1	1
beautiful night football world cup semi finals qatar	positive	1	1

Athena

pred_full	
text	string
clean	string
sentiment	string
label	double
prediction	double

```sql
CREATE TABLE incorrect_words AS
    (SELECT word, sentiment, label, prediction, is_correct
    FROM (
    SELECT split(clean, ' ') AS words,
    sentiment,
    label,
    prediction,
    is_correct
    FROM predictions
    ) t1
CROSS JOIN UNNEST(words) AS t2(word)
WHERE word NOT LIKE ''
AND is_correct = 'incorrect');
```

football	
id	bigint
user_name	string
user_screen_name	string
text	string
follower_count	int
location	string
created_at	string
sentiment	string

```sql
CREATE TABLE words_pos AS
    (SELECT word, sentiment
    FROM (
            SELECT split(clean, ' ') as words,
                sentiment
            FROM pred_full
    ) t1
CROSS JOIN UNNEST(words) AS t2(word)
WHERE word NOT LIKE ''
AND sentiment = 'positive')
;
```

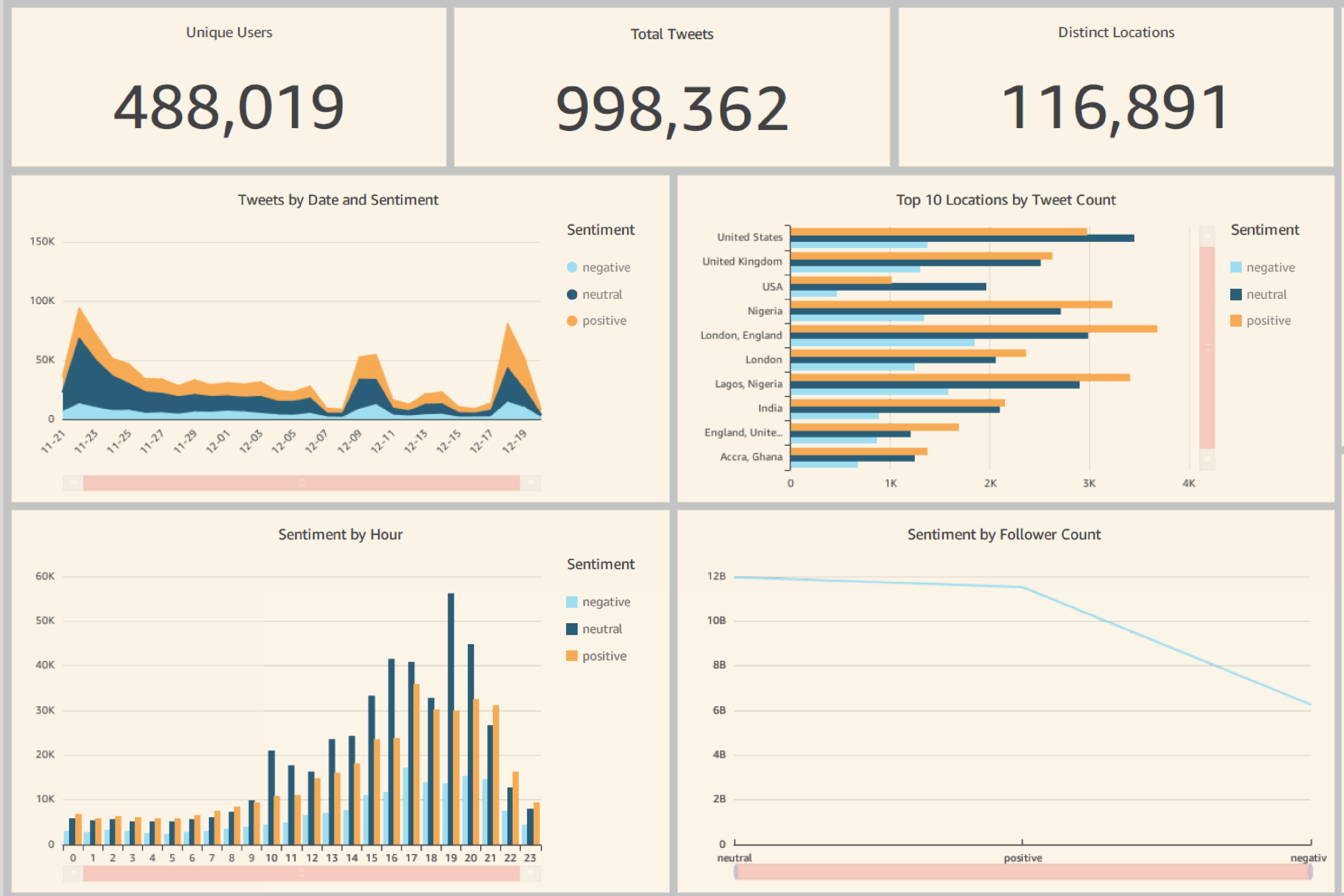
```sql
CREATE TABLE predictions AS
    (SELECT *,
    CASE WHEN label = prediction THEN 'correct'
    ELSE 'incorrect'
    END AS is_correct
    FROM pred_full)
;
```

```sql
CREATE TABLE clean AS
    (SELECT
    user_name,
    follower_count,
    location,
    text,
    sentiment,
    date_format(from_iso8601_timestamp(created_at), '%m-%d') AS month_day,
    extract(month from from_iso8601_timestamp(created_at)) AS month,
    extract(day from from_iso8601_timestamp(created_at)) AS day,
    extract(dow from from_iso8601_timestamp(created_at)) AS day_of_week,
    extract(hour from from_iso8601_timestamp(created_at)) AS hour
FROM football)
;
```

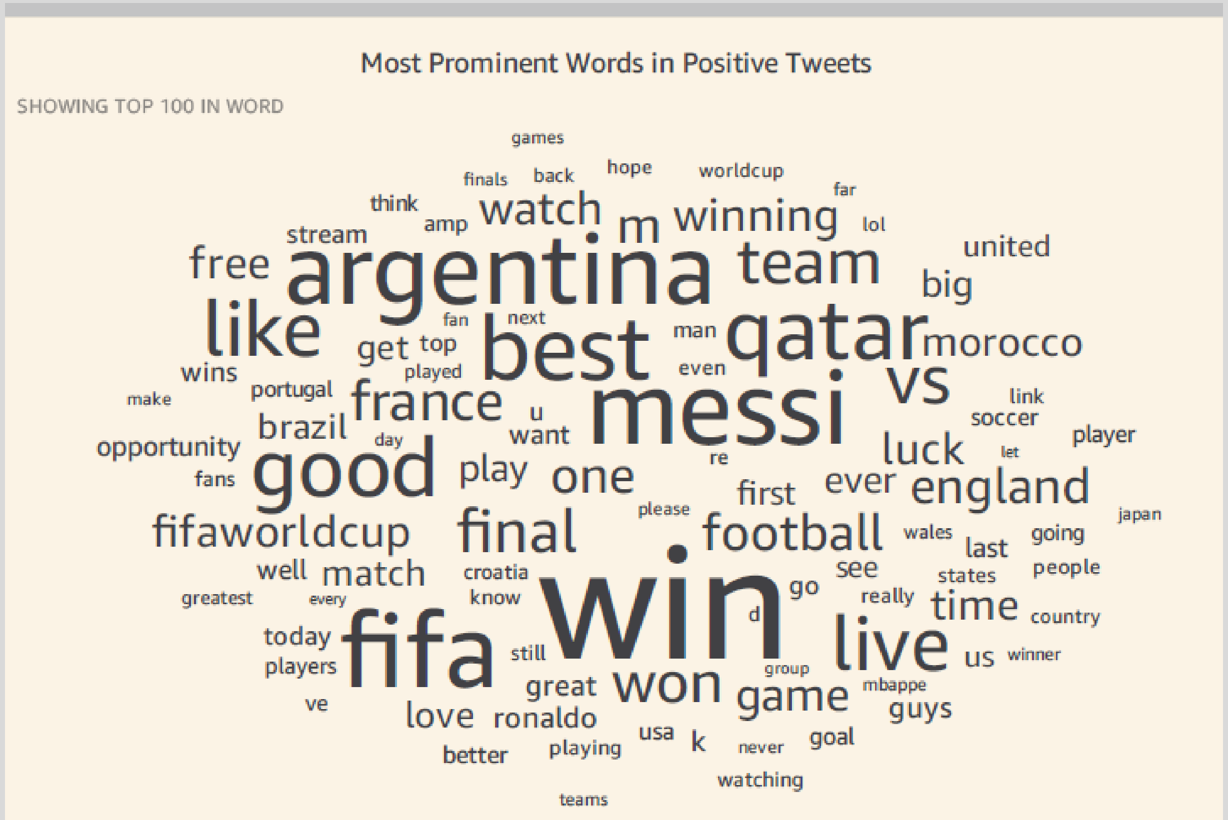
```sql
## Tables created to use in Athena
-- 1. clean
-- 2. words
-- 3. words_pos
-- 4. words_neg
-- 5. predictions
-- 6. incorrect_words
```

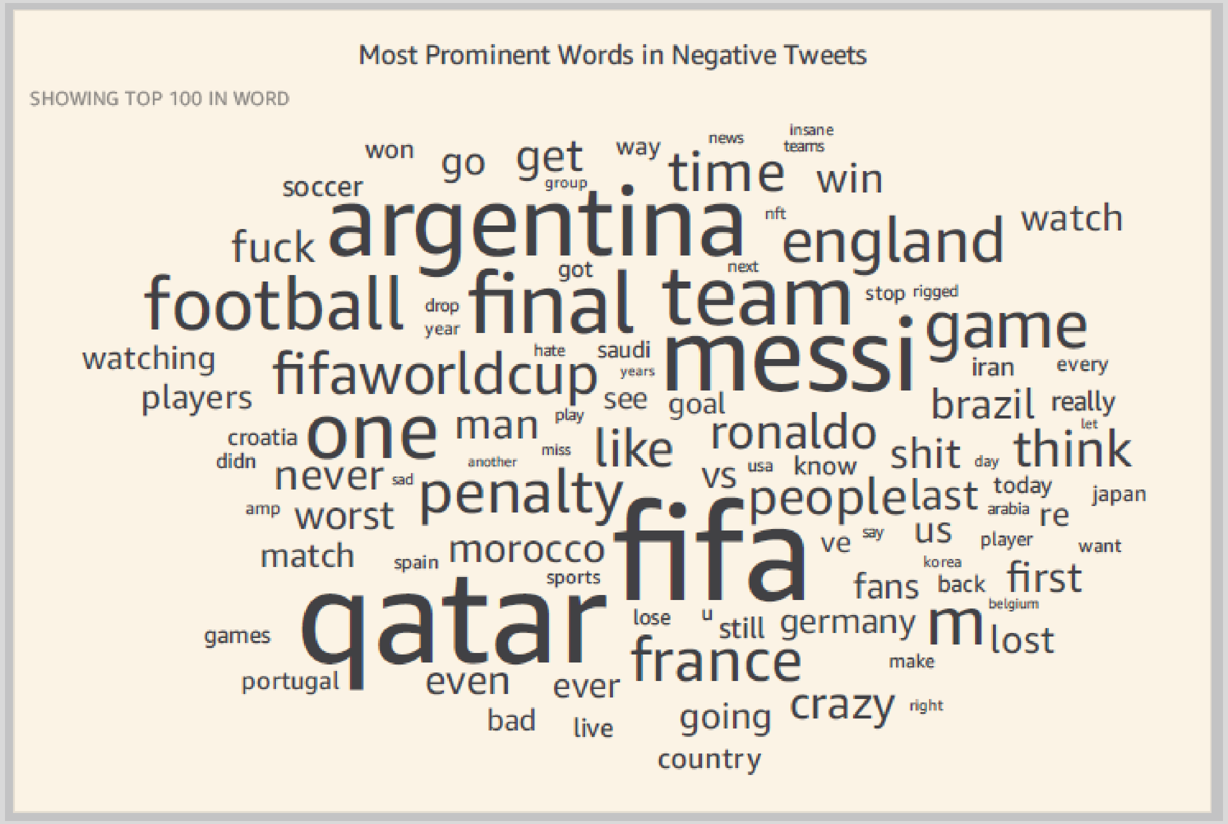
QuickSight

Dataset Analysis

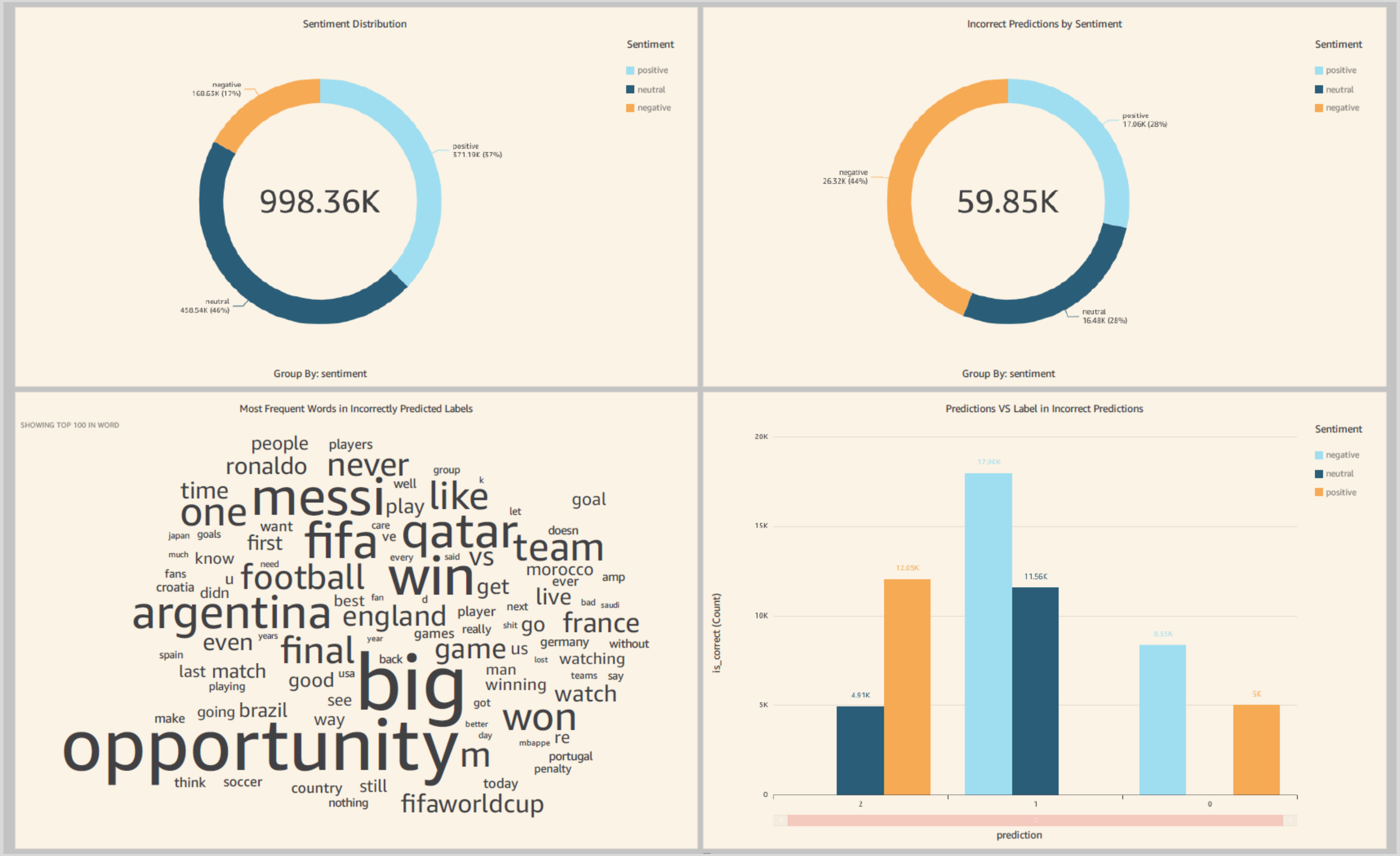


QuickSight – Word Cloud





QuickSight - Prediction Analysis



Challenges

- Adjusting to syntax
- Community Edition Databricks computing power
- New environment, lack of understanding of Databricks troubleshooting

Next Steps

- Trying out paid Databricks editions
- Testing other tree-based models
- Getting better understanding of cloud computing