Chapter 11: NLP

Ex1: Ham vs Spam

Requirement: Build a spam filter. Use the various NLP tools and a new classifier, Naive Bayes, to predict if one email is ham or spam.

- Dataset: UCI Repository SMS Spam Detection: https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection (https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection)

In [1]:
```
import findspark
findspark.init()
```

In [2]:
```
from pyspark.sql import SparkSession
```

In [3]:
```
spark = SparkSession.builder.appName('nlp').getOrCreate()
```

In [4]:
```
data = spark.read.csv("smsspamcollection/SMSSpamCollection",
                      inferSchema=True,
                      sep='\t')
```

In [5]:
```
data = data.withColumnRenamed('_c0','class').withColumnRenamed('_c1','text')
```

In [6]:
```
data.show(5)
```

```
+-----+--------------------+
|class|                text|
+-----+--------------------+
|  ham|Go until jurong p...|
|  ham|Ok lar... Joking ...|
| spam|Free entry in 2 a...|
|  ham|U dun say so earl...|
|  ham|Nah I don't think...|
+-----+--------------------+
only showing top 5 rows
```

Clean and Prepare the Data

** Create a new length feature: **

In [7]:
```
from pyspark.sql.functions import length
```

In [8]:
```
data = data.withColumn('length',length(data['text']))
```

In [9]:
```
data.show(5)
```

```
+-----+--------------------+------+
|class|                text|length|
+-----+--------------------+------+
|  ham|Go until jurong p...|   111|
|  ham|Ok lar... Joking ...|    29|
| spam|Free entry in 2 a...|   155|
|  ham|U dun say so earl...|    49|
|  ham|Nah I don't think...|    61|
+-----+--------------------+------+
only showing top 5 rows
```

In [10]:
```
# Pretty Clear Difference
data.groupby('class').mean().show()
```

```
+-----+-----------------+
|class|      avg(length)|
+-----+-----------------+
|  ham|71.45431945307645|
| spam|138.6706827309237|
+-----+-----------------+
```

Feature Transformations

In [11]:
```
from pyspark.ml.feature import Tokenizer,StopWordsRemover
from pyspark.ml.feature import CountVectorizer, IDF, StringIndexer
tokenizer = Tokenizer(inputCol="text", outputCol="token_text")
stopremove = StopWordsRemover(inputCol='token_text',outputCol='stop_tokens')
count_vec = CountVectorizer(inputCol='stop_tokens',outputCol='c_vec')
idf = IDF(inputCol="c_vec", outputCol="tf_idf")
ham_spam_to_num = StringIndexer(inputCol='class',outputCol='label')
```

In [12]:
```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.linalg import Vector
```

In [13]:
```
clean_up = VectorAssembler(inputCols=['tf_idf','length'],
                           outputCol='features')
```

The Model

We'll use Naive Bayes, but feel free to play around with this choice!

In [14]:
```
from pyspark.ml.classification import NaiveBayes
```

In [15]:
```
# Use defaults
nb = NaiveBayes()
```

Pipeline

In [16]:
```
from pyspark.ml import Pipeline
```

In [17]:
```
data_prep_pipe = Pipeline(stages=[ham_spam_to_num,
                                  tokenizer,
                                  stopremove,
                                  count_vec,
                                  idf,
                                  clean_up])
```

In [18]:
```
cleaner = data_prep_pipe.fit(data)
```

In [19]:
```
clean_data = cleaner.transform(data)
```

Training and Evaluation!

In [20]:
```
clean_data = clean_data.select(['label','features'])
```

In [21]:
```
clean_data.show(10)
```

```
+-----+--------------------+
|label|            features|
+-----+--------------------+
|  0.0|(13424,[7,11,31,6...|
|  0.0|(13424,[0,24,297,...|
|  1.0|(13424,[2,13,19,3...|
|  0.0|(13424,[0,70,80,1...|
|  0.0|(13424,[36,134,31...|
|  1.0|(13424,[10,60,139...|
|  0.0|(13424,[10,53,103...|
|  0.0|(13424,[125,184,4...|
|  1.0|(13424,[1,47,118,...|
|  1.0|(13424,[0,1,13,27...|
+-----+--------------------+
only showing top 10 rows
```

In [22]:
```
(training,testing) = clean_data.randomSplit([0.7,0.3])
```

In [23]:
```
spam_predictor = nb.fit(training)
```

In [24]:
```
data.printSchema()
```

```
root
 |-- class: string (nullable = true)
 |-- text: string (nullable = true)
 |-- length: integer (nullable = true)
```

In [25]:
```
test_results = spam_predictor.transform(testing)
```

In [26]:
```
test_results.show(10)
```

```
+-----+--------------------+--------------------+--------------------+---------
-+
|label|            features|       rawPrediction|         probability|predictio
n|
+-----+--------------------+--------------------+--------------------+---------
-+
|  0.0|(13424,[0,1,2,7,8...|[-805.18281628272...|[1.0,1.5026888754...|       0.
0|
|  0.0|(13424,[0,1,3,9,1...|[-571.08015050035...|[0.99999999999989...|       0.
0|
|  0.0|(13424,[0,1,7,8,1...|[-1151.0845440365...|[1.0,1.7344215452...|       0.
0|
|  0.0|(13424,[0,1,9,14,...|[-560.99061513047...|[1.0,1.1693266509...|       0.
0|
|  0.0|(13424,[0,1,9,14,...|[-560.99061513047...|[1.0,1.1693266509...|       0.
0|
|  0.0|(13424,[0,1,12,33...|[-443.53614103103...|[1.0,4.2387365649...|       0.
0|
|  0.0|(13424,[0,1,18,20...|[-830.74555602901...|[1.0,4.1260268479...|       0.
0|
|  0.0|(13424,[0,1,21,27...|[-757.45725664880...|[1.0,4.2310444011...|       0.
0|
|  0.0|(13424,[0,1,21,27...|[-1025.2055750518...|[1.0,9.2583979062...|       0.
0|
|  0.0|(13424,[0,1,24,31...|[-343.57478257252...|[1.0,7.9081642069...|       0.
0|
+-----+--------------------+--------------------+--------------------+---------
-+
only showing top 10 rows
```

In [27]:
```
test_results.groupBy("label", "prediction").count().show()
```

```
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|  1.0|       1.0|  217|
|  0.0|       1.0|  152|
|  1.0|       0.0|    8|
|  0.0|       0.0| 1323|
+-----+----------+-----+
```

In [28]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

In [29]:
```
acc_eval = MulticlassClassificationEvaluator()
acc = acc_eval.evaluate(test_results)
print("Accuracy of model at predicting spam was: {}".format(acc))
```

```
Accuracy of model at predicting spam was: 0.9148755595026191
```

Not bad considering we're using straight math on text data! Try switching out the classification models! Or even try to come up with other engineered features!