Ham vs Spam Detector with Naive Bayes

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes

In [2]:
```
import findspark
findspark.init()
```

In [3]:
```
from pyspark.sql import SparkSession
```

In [4]:
```
spark = SparkSession.builder.appName('spamdetector').master('local[*]').getOrCreate()
```

Load and Explore Data

In [5]:
```
data = spark.read.csv('data/SMSSpamCollection', inferSchema=True, sep='\t')
```

In [6]:
```
data.show(9)
```

```
+----+--------------------+
| _c0|                 _c1|
+----+--------------------+
| 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...|
|spam|FreeMsg Hey there...|
| ham|Even my brother i...|
| ham|As per your reque...|
|spam|WINNER!! As a val...|
+----+--------------------+
only showing top 9 rows
```

In [7]:
```
data = data.withColumnRenamed('_c0', 'class').withColumnRenamed('_c1', 'text')
```

In [8]:
```
# we have sentences that are labelled as ham or spam
data.show()
```

```
+-----+--------------------+
|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...|
| spam|FreeMsg Hey there...|
|  ham|Even my brother i...|
|  ham|As per your reque...|
| spam|WINNER!! As a val...|
| spam|Had your mobile 1...|
|  ham|I'm gonna be home...|
| spam|SIX chances to wi...|
| spam|URGENT! You have ...|
|  ham|I've been searchi...|
|  ham|I HAVE A DATE ON ...|
```

```
| spam|XXXMobileMovieClu...|
|  ham|Oh k...i'm watchi...|
|  ham|Eh u remember how...|
|  ham|Fine if that□s th...|
| spam|England v Macedon...|
+-----+--------------------+
only showing top 20 rows
```

In [9]:
```
from pyspark.sql.functions import length
```

In [10]:
```
data = data.withColumn('length', length(data['text']))
```

In [11]:
```
data.show()
```

```
+-----+--------------------+------+
|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|
| spam|FreeMsg Hey there...|   147|
|  ham|Even my brother i...|    77|
|  ham|As per your reque...|   160|
| spam|WINNER!! As a val...|   157|
| spam|Had your mobile 1...|   154|
|  ham|I'm gonna be home...|   109|
| spam|SIX chances to wi...|   136|
| spam|URGENT! You have ...|   155|
|  ham|I've been searchi...|   196|
|  ham|I HAVE A DATE ON ...|    35|
| spam|XXXMobileMovieClu...|   149|
|  ham|Oh k...i'm watchi...|    26|
|  ham|Eh u remember how...|    81|
|  ham|Fine if that□s th...|    56|
| spam|England v Macedon...|   155|
+-----+--------------------+------+
only showing top 20 rows
```

In [12]:
```
# on average we see that spam is almost twice as long as ham!
data.groupBy('class').mean().show()
```

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

Pre-processing Text

In [13]:
```
from pyspark.ml.feature import (
    Tokenizer, StopWordsRemover, CountVectorizer, IDF, StringIndexer)
```

In [14]:
```
# tokenize sentence
tokenizer = Tokenizer(inputCol='text', outputCol='token_text')
# remove stop words
stop_remove = StopWordsRemover(inputCol='token_text', outputCol='stop_tokens')
```

```
# convert each sentence to vector of token counts
# this vector has length that is size of vocab
count_vec = CountVectorizer(inputCol='stop_tokens', outputCol='c_vec')
# calculate the inverse document frequency (IDF) from the count vectors
idf = IDF(inputCol='c_vec', outputCol='idf')
# convert class label to numeric target
ham_spam_to_numeric = StringIndexer(inputCol='class', outputCol='label')
```

In [15]:
```
from pyspark.ml.feature import VectorAssembler
```

In [16]:
```
clean_up = VectorAssembler(inputCols=['c_vec', 'idf', 'length'], outputCol='features')
```

In [17]:
```
from pyspark.ml.classification import NaiveBayes
```

In [18]:
```
nb = NaiveBayes()
```

In [19]:
```
from pyspark.ml import Pipeline
```

In [20]:
```
data_prep_pipe = Pipeline(stages=[ham_spam_to_numeric, tokenizer, stop_remove, count_ve
```

In [21]:
```
cleaner = data_prep_pipe.fit(data)
```

In [22]:
```
clean_data = cleaner.transform(data)
```

In [23]:
```
clean_data = clean_data.select('label', 'features')
```

In [23]:
```
clean_data.show()
```

```
+-----+--------------------+
|label|            features|
+-----+--------------------+
|  0.0|(26847,[7,11,31,6...|
|  0.0|(26847,[0,24,297,...|
|  1.0|(26847,[2,13,19,3...|
|  0.0|(26847,[0,70,80,1...|
|  0.0|(26847,[36,134,31...|
|  1.0|(26847,[10,60,139...|
|  0.0|(26847,[10,53,103...|
|  0.0|(26847,[125,184,4...|
|  1.0|(26847,[1,47,118,...|
|  1.0|(26847,[0,1,13,27...|
|  0.0|(26847,[18,43,120...|
|  1.0|(26847,[8,17,37,8...|
|  1.0|(26847,[13,30,47,...|
|  0.0|(26847,[39,96,217...|
|  0.0|(26847,[552,1697,...|
|  1.0|(26847,[30,109,11...|
|  0.0|(26847,[82,214,47...|
|  0.0|(26847,[0,2,49,13...|
|  0.0|(26847,[0,74,105,...|
|  1.0|(26847,[4,30,33,5...|
+-----+--------------------+
only showing top 20 rows
```

Train Naive Bayes Classifier

In [25]:
```
training, test = clean_data.randomSplit([0.7, 0.3])
```

In [26]:
```
spam_detector = nb.fit(training)
```

In [27]:
```
data.printSchema()
```

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

In [28]:
```
test_results = spam_detector.transform(test)
```

Evaluate Results

In [29]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

In [30]:
```
acc_eval = MulticlassClassificationEvaluator()
```

In [31]:
```
acc = acc_eval.evaluate(test_results)
```

In [32]:
```
acc
```

Out[32]:
```
0.9460545026072913
```

In [ ]:

In [ ]:

In [ ]: