# NLP Code Along

For this code along we will build a spam filter! We'll use the various NLP tools we learned about as well as a new classifier, Naive Bayes.

We'll use a classic dataset for this - UCI Repository SMS Spam Detection:

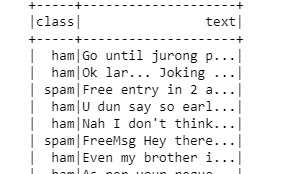
**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.appName('nlp').getOrCreate()

data = spark.read.csv("smsspamcollection/SMSSpamCollection",inferSchema=**True**,sep='**\t**')

data = data.withColumnRenamed('\_c0','class').withColumnRenamed('\_c1','text')

data.show()



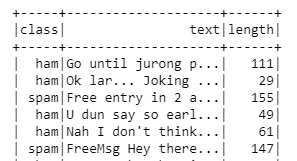
## Clean and Prepare the Data

**Create a new length feature:**

**from** **pyspark.sql.functions** **import** length

data = data.withColumn('length',length(data['text']))

data.show()



*# Pretty Clear Difference*

data.groupby('class').mean().show()

**Feature Transformations**

**from** **pyspark.ml.feature** **import** Tokenizer,StopWordsRemover, 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')

**from** **pyspark.ml.feature** **import** VectorAssembler

**from** **pyspark.ml.linalg** **import** Vector

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!

**from** **pyspark.ml.classification** **import** NaiveBayes

*# Use defaults*

nb = NaiveBayes()

**Pipeline**

**from** **pyspark.ml** **import** Pipeline

data\_prep\_pipe = Pipeline(stages=[ham\_spam\_to\_num,tokenizer,stopremove,count\_vec,idf,clean\_up])

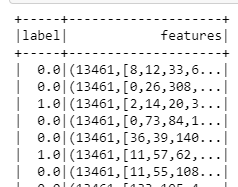
cleaner = data\_prep\_pipe.fit(data)

clean\_data = cleaner.transform(data)

### Training and Evaluation!

clean\_data = clean\_data.select(['label','features'])

clean\_data.show()



(training,testing) = clean\_data.randomSplit([0.7,0.3])

spam\_predictor = nb.fit(training)

data.printSchema()

root

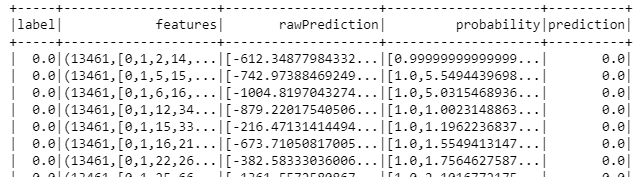
|-- label: string (nullable = true)

|-- text: string (nullable = true)

|-- length: integer (nullable = true)

test\_results = spam\_predictor.transform(testing)

test\_results.show()



**from** **pyspark.ml.evaluation** **import** MulticlassClassificationEvaluator

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: 0.9248020435242028

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!