[Twitter Airlines Analysis](http://122.165.140.148/terasql/deep-insights-dsci/-/tree/patch-1/Twitter%20Sentiment%20Analysis)

Data Preparation

Data is downloaded from <https://www.kaggle.com/crowdflower/twitter-airline-sentiment>

A sentiment analysis job about the problems of each major U.S. airline. Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as "late flight" or "rude service").

It contains the following fields:

1. tweet\_id
2. airline\_sentiment
3. airline\_sentiment\_confidence
4. negativereason
5. negativereason\_confidence
6. airline
7. airline\_sentiment\_gold
8. name
9. negativereason\_gold
10. retweet\_count
11. text
12. tweet\_coord
13. tweet\_created
14. tweet\_location

15. user\_timezone

Firstly, we load the data.

val df = spark.read.

format("csv")

.option("header","true")

.option("inferSchema","true")

.load("/home/harsh/Desktop/twitter airlines/Tweets.csv")

Selecting only required fields:

val raw = df.select($"airline\_sentiment".alias("label"),$"text".alias("tweet"))

Removing null values :

val nr = raw.na.drop()

Converting categorical features to numerical :

val indexer = new StringIndexer()

.setInputCol("label")

.setOutputCol("labelIndex")

val indexed = indexer.fit(nr).transform(nr)

Making a user defined function for pre-processing (removing special characters, emojis, website links) :

def lo(d:String) :String = { d.replace("\"","").toLowerCase()

.replaceAll("\n", "")

.replaceAll("rt\\s+", "")

.replaceAll("\\s+@\\w+", "")

.replaceAll("@\\w+", "")

.replaceAll("\\s+#\\w+", "")

.replaceAll("#\\w+", "")

.replaceAll("(?:https?|http?)://[\\w/%.-]+", "")

.replaceAll("(?:https?|http?)://[\\w/%.-]+\\s+", "")

.replaceAll("(?:https?|http?)//[\\w/%.-]+\\s+", "")

.replaceAll("(?:https?|http?)//[\\w/%.-]+", "")

.replaceAll("[^\u0000-\uFFFF]","")

.replaceAll("(\u00a9|\u00ae|[\u2000-\u3300]|\ud83c[\ud000-\udfff]|\ud83d[\ud000-\udfff]|\ud83e[\ud000-\udfff])","")

.trim()

}

val lco = udf(lo \_)

Pre-processing data using udf :

val f = indexed.select($"label", $"labelIndex", lco($"tweet").alias("tweet"))

After loading and pre-processing, we need to convert tweets into feature vectors.

val tokenizer = new Tokenizer().setInputCol("tweet").setOutputCol("words")

val wordsData = tokenizer.transform(f)

val hashingTF = new HashingTF()

.setInputCol("words").setOutputCol("rawFeatures").setNumFeatures(10000)

val featurizedData = hashingTF.transform(wordsData)

val idf = new IDF().setInputCol("rawFeatures").setOutputCol("features")

val idfModel = idf.fit(featurizedData)

val rescaledData = idfModel.transform(featurizedData)

Then we split the transformed data into two subsets i.e. training and test(ratio 0.8:0.2)

val Array(training, test) = rescaledData.randomSplit(Array[Double](0.8,0.2))

Model Selection and Model Tuning

We tried Logistic Regression and MultiLayer Perceptron for classification.

val lr = new LogisticRegression()

.setMaxIter(10)

.setRegParam(0.01)

.setLabelCol("labelIndex")

.setElasticNetParam(0.5)

val layers = Array[Int](20,15, 10, 3)

val trainer = new MultilayerPerceptronClassifier() .setLayers(layers)

.setLabelCol("labelIndex")

val model = lr.fit(training)

val model2 = trainer.fit(training)

val lr\_predictions = model.transform(test)

val mlp\_predictions = model2.transform(test)

Conclusion

We evaluated accuracy for Logistic Regression and MultiLayer Perceptron using MultiClassClassification Evaluator and got 77 % accuracy for Logistic Regression and 74 % for MultiLayer Perceptron.

val evaluator = new MulticlassClassificationEvaluator()

.setLabelCol("label")

.setPredictionCol("prediction")

.setMetricName("accuracy")

val lr\_accuracy = evaluator.evaluate(lr\_predictions)

val mlp\_accuracy = evaluator.evaluate(mlp\_predictions)