Ben Gutstein
DE300
Homework 3 Write up

Part 1:

The objective here was to compute TF-IDF scores from the agnews_clean.csv file, using PySpark. The csv contains document IDs and filtered arrays of words used in each document. I first needed to read and parse the csv using Spark DataFrame with inferSchema=True. I then converted the "filtered" column from a JSON string into an array of strings using "F.from_json". Next, I created an rdd of (doc_id, word) pairs for term-level analysis of all the documents. Then, I counted term frequency per document using reduceByKey and calculated each document's length to normalize term frequency. I collected the document frequency for each unique word using the variable doc_freq. I then computed the IDF using the formula given to us in the homework instructions. The next step was to multiply TF and IDF to get the final TF-IDF scores and lastly, to group the TF-IDF scores by document to output the top 5 terms per each document.

```
TF-IDF scores for first 5 documents:
[Stage 12:>                                                 (0 + 9) / 9]
Doc 0: [('cynics', 0.563734318747707), ('wall', 0.5115985326511431), ('claw', 0.499114829314058), ('dwindling', 0.4572386180709258), ('sellers', 0.4468379768438066)]
Doc 9: [('cynics', 0.5340640914451961), ('wall', 0.48467229409055657), ('claw', 0.47284562777121286), ('dwindling', 0.43317342764614025), ('sellers', 0.4233201885888694)]
Doc 18: [('deficit', 0.540640233790213), ('swells', 0.4456552560521169), ('trade', 0.4080689115854956), ('8bn', 0.39631754501941335), ('imports', 0.31527891489726495)]
Doc 27: [('fall', 0.43167912777209483), ('shares', 0.386492653524941), ('tumble', 0.3419973653042592), ('quarter', 0.3411920291869863), ('disappointing', 0.29431935936888143)]
Doc 36: [('google', 0.34612714944621187), ('secrecy', 0.3432093194552543), ('confusing', 0.3299462238286493), ('submitted', 0.28716139264401674), ('news', 0.27869755632552773)]
```

Part 2:

The objective in part 2 was to evaluate an SVM model's loss and prediction accuracy using given weights, bias and a labeled dataset. The first step to making the loss function was making the soft-margin SVM loss equation that we were given from the homework sheet. Lambda was set to 1.0. The weights (w.csv) and bias (bias.csv) were loaded in and used them with the feature matrix and labels from our other csv. Final loss was computed as a scalar. Next, the prediction function was written by implementing the classifier given on the sheet. I applied it to all data points in order to predict the labels. The results are shown below. We see the SVM loss and the prediction that the function made using the given classifier.

```
loss = loss_SVM(w, bias, data_svm, data_svm.iloc[:, -1])
print("SVM Loss:", loss)
```

✓ 0.0s

```
SVM Loss: 1.0029403834857522
```

```
y_pred = predict_SVM(w, bias, data_svm)
print("Predictions:\n", y_pred[:10])
```

✓ 0.0s

```
Predictions:
 [-1. -1. -1.  1. -1.  1. -1. -1.  1. -1.]
```