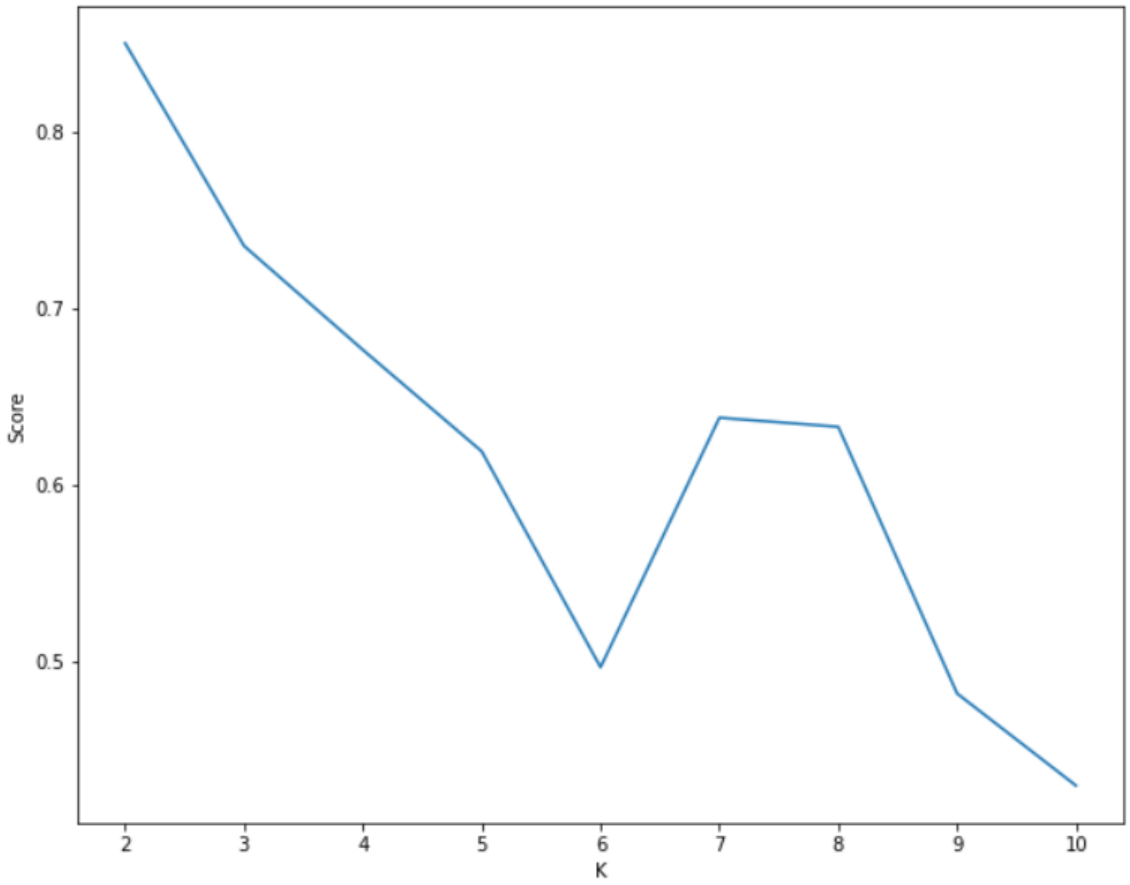
Humza Salman
mhs180007

Homework 4

Question 1: Clustering

a) **Plot of Silhouette Score vs K for K-Means:**



The silhouette score is a measure of clustering technique. The closer we are to a value of 1 the more the inter-cluster distance is maximized and the more the intra-cluster distance is minimized. Values closer to 0 indicate that distance between clusters is not significant. The silhouette score also ranges until -1, however we do not see that in our graph. We see that k=2 gives the highest silhouette score, however we know our ground truth indicates that we have 3 varying classes. Instead, we should pick k=3 since it is a local maximum considering our context of our data having 3 classes. K=1 is not chosen as a starting point since it throws an error when passing into the KMeans model from pyspark.

K=3 is chosen as our final answer.

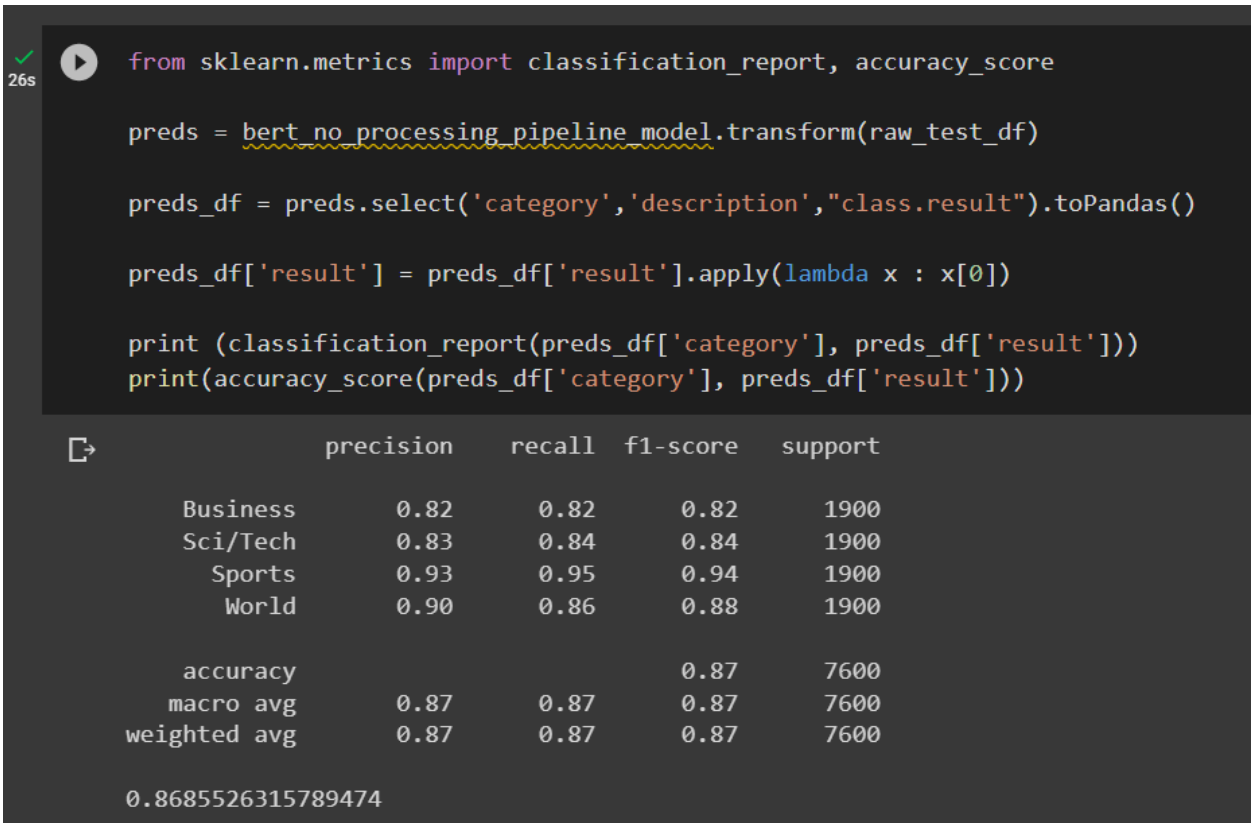
b) Since we chose K=3 from part (a) we get a **silhouette score of 0.73 for KMeans**. Running **Gaussian Mixture for K=3 we get a silhouette score of**

0.43. The **KMeans is a better model in this scenario** since the silhouette score indicates that the inter-cluster distance is maximized, and the intra-cluster distance is minimized to a further extent than the Gaussian Mixture model. This means that with K=3 the clusters formed by KMeans are better at grouping data points together.

Question 2: Spark NLP

a) **For my no preprocessing pipeline I chose to do the following:**
Document Assembler -> Tokenizer -> Bert Embeddings -> Sentence Embeddings -> classifierDL (5 epochs, 0.001 learning rate)

Tokenizing was essential in this case since the BertEmbeddings expect both the document and the token. With this model I saw the **accuracy to be 0.8685 or 86.85% on the test dataset**.



```
from sklearn.metrics import classification_report, accuracy_score

preds = bert_no_processing_pipeline_model.transform(raw_test_df)

preds_df = preds.select('category','description',"class.result").toPandas()

preds_df['result'] = preds_df['result'].apply(lambda x : x[0])

print (classification_report(preds_df['category'], preds_df['result']))
print(accuracy_score(preds_df['category'], preds_df['result']))
```

```
              precision    recall  f1-score   support

    Business       0.82      0.82      0.82      1900
    Sci/Tech       0.83      0.84      0.84      1900
      Sports       0.93      0.95      0.94      1900
       World       0.90      0.86      0.88      1900

    accuracy                           0.87      7600
   macro avg       0.87      0.87      0.87      7600
weighted avg       0.87      0.87      0.87      7600

0.8685526315789474
```

b) **The following shows varying preprocessing pipelines:-**

Only Stopwords pipeline:
Document Assembler -> Tokenizer -> Normalizer -> Stopwords Cleaner -> Bert Embeddings -> Sentence Embeddings -> classifierDL (5 epochs, 0.001 learning

rate)

I saw in the given references they were using a normalizer to aid in preprocessing the data, so I implemented it as well. With this model I saw the **accuracy to be 0.87 or 87% on the test dataset.**

```
from sklearn.metrics import classification_report, accuracy_score

preds = bert_stopwords_pipeline_model.transform(raw_test_df)

preds_df = preds.select('category','description',"class.result").toPandas()

preds_df['result'] = preds_df['result'].apply(lambda x : x[0])

print (classification_report(preds_df['category'], preds_df['result']))
print(accuracy_score(preds_df['category'], preds_df['result']))
```

```
              precision    recall  f1-score   support

    Business       0.82      0.81      0.82      1900
    Sci/Tech       0.83      0.86      0.84      1900
      Sports       0.94      0.95      0.95      1900
       World       0.89      0.86      0.88      1900

    accuracy                           0.87      7600
   macro avg       0.87      0.87      0.87      7600
weighted avg       0.87      0.87      0.87      7600

0.87
```

Only Lemmatizer pipeline:
Document Assembler -> Tokenizer -> Normalizer -> Lemmatizer -> Bert Embeddings -> Sentence Embeddings -> classifierDL (5 epochs, 0.001 learning rate)

I saw in the given references they were using a normalizer to aid in preprocessing the data, so I implemented it as well. With this model I saw the **accuracy to be 0.8639 or 86.39% on the test dataset.**

```
from sklearn.metrics import classification_report, accuracy_score

preds = bert_lemma_pipeline_model.transform(raw_test_df)

preds_df = preds.select('category','description',"class.result").toPandas()

preds_df['result'] = preds_df['result'].apply(lambda x : x[0])

print (classification_report(preds_df['category'], preds_df['result']))
print(accuracy_score(preds_df['category'], preds_df['result']))
```

```
              precision    recall  f1-score   support

    Business       0.82      0.80      0.81      1900
    Sci/Tech       0.81      0.85      0.83      1900
      Sports       0.93      0.95      0.94      1900
       World       0.90      0.85      0.87      1900

    accuracy                           0.86      7600
   macro avg       0.86      0.86      0.86      7600
weighted avg       0.86      0.86      0.86      7600

0.8639473684210527
```

Stopwords and Lemmatizer pipeline:
Document Assembler -> Tokenizer -> Normalizer -> Stopwords Cleaner -> Lemmatizer -> Bert Embeddings -> Sentence Embeddings -> classifierDL (5 epochs, 0.001 learning rate)

I saw in the given references they were using a normalizer to aid in preprocessing the data, so I implemented it as well. This model also uses stopwords from which the tokens are fed to be lemmatized and then into the Bert Embeddings. With this model I saw the **accuracy to be 0.8681 or 86.81% on the test dataset**.

```
from sklearn.metrics import classification_report, accuracy_score

preds = bert_stopwords_lemma_pipeline_model.transform(raw_test_df)

preds_df = preds.select('category','description',"class.result").toPandas()

preds_df['result'] = preds_df['result'].apply(lambda x : x[0])

print (classification_report(preds_df['category'], preds_df['result']))
print(accuracy_score(preds_df['category'], preds_df['result']))
```

```
              precision    recall  f1-score   support

    Business       0.81      0.82      0.81      1900
    Sci/Tech       0.82      0.85      0.84      1900
      Sports       0.95      0.94      0.95      1900
       World       0.89      0.86      0.88      1900

    accuracy                           0.87      7600
   macro avg       0.87      0.87      0.87      7600
weighted avg       0.87      0.87      0.87      7600

0.8681578947368421
```

Pipeline that performs the best:

Only Stopwords Cleaner. This pipeline performed the best in terms of accuracy - we get 87% accuracy compared to the slightly lower accuracy scores we got for other models. I am not sure why only stopwords performs the best. My assumption would be that Normalizer performs a lot of preprocessing steps such as lowercasing, removing punctuation, etc. by itself and it may be doing stopwords on other models as well. Lemmatizing may cause the model to not utilize the full extent of Bert Embeddings capabilities and its vocabulary. I think overall the I would still choose the stopwords and lemmatized model to be considered the best since it involves more preprocessing steps and is only marginally worse by 0.19% in accuracy on the test set, so I will be using this model for the RoBerta Embeddings pipeline.

c) **Best pipeline with Roberta**
Document Assembler -> Tokenizer -> Normalizer -> Stopwords Cleaner -> Lemma -> RoBerta Embeddings -> Sentence Embeddings -> classifierDL (5 epochs, 0.001 learning rate)

I chose to use the pipeline with a tokenizer, normalizer, stopwords cleaner, and a lemmatizer that is fed into the RoBerta Embeddings to create sentence embeddings that are fed into the classifierDL model with the same

hyperparameters. With this **model I saw the accuracy to be 0.8777 or 87.77%** which is an increase from what we saw from any of the Bert Embeddings models.

For Bert Embeddings I used the pretrained “small_bert_L4_256” model with the language parameter set to English and for RoBerta Embeddings I used the pretrained “roberta_embeddings_distilroberta_base” model with the language parameter set to English**. RoBerta’s higher accuracy score can be reasoned by its architecture being more refined than Bert’s since RoBerta considers a larger and diverse corpus and trains with dynamic masking making it more robust**. This could provide it a slight edge in these classification tasks when it comes to longer sequences.

RoBerta Embeddings performed better than Bert Embeddings.

```
from sklearn.metrics import classification_report, accuracy_score

preds = roberta_pipeline_model.transform(raw_test_df)

preds_df = preds.select('category','description',"class.result").toPandas()

preds_df['result'] = preds_df['result'].apply(lambda x : x[0])

print (classification_report(preds_df['category'], preds_df['result']))
print(accuracy_score(preds_df['category'], preds_df['result']))
```

```
              precision    recall  f1-score   support

    Business       0.83      0.83      0.83      1900
    Sci/Tech       0.84      0.86      0.85      1900
      Sports       0.93      0.97      0.95      1900
       World       0.92      0.85      0.88      1900

    accuracy                           0.88      7600
   macro avg       0.88      0.88      0.88      7600
weighted avg       0.88      0.88      0.88      7600

0.8777631578947368
```