IR BONUS

Advanced Text Classification and Model Evaluation

Rachelli Adler 213836687
Shiffy Schiner 328949714
Github: https://github.com/shiffy01/IR_Bonus.git
Course ID: 157125.21.5785
Date: 11/02/2025

Part A:

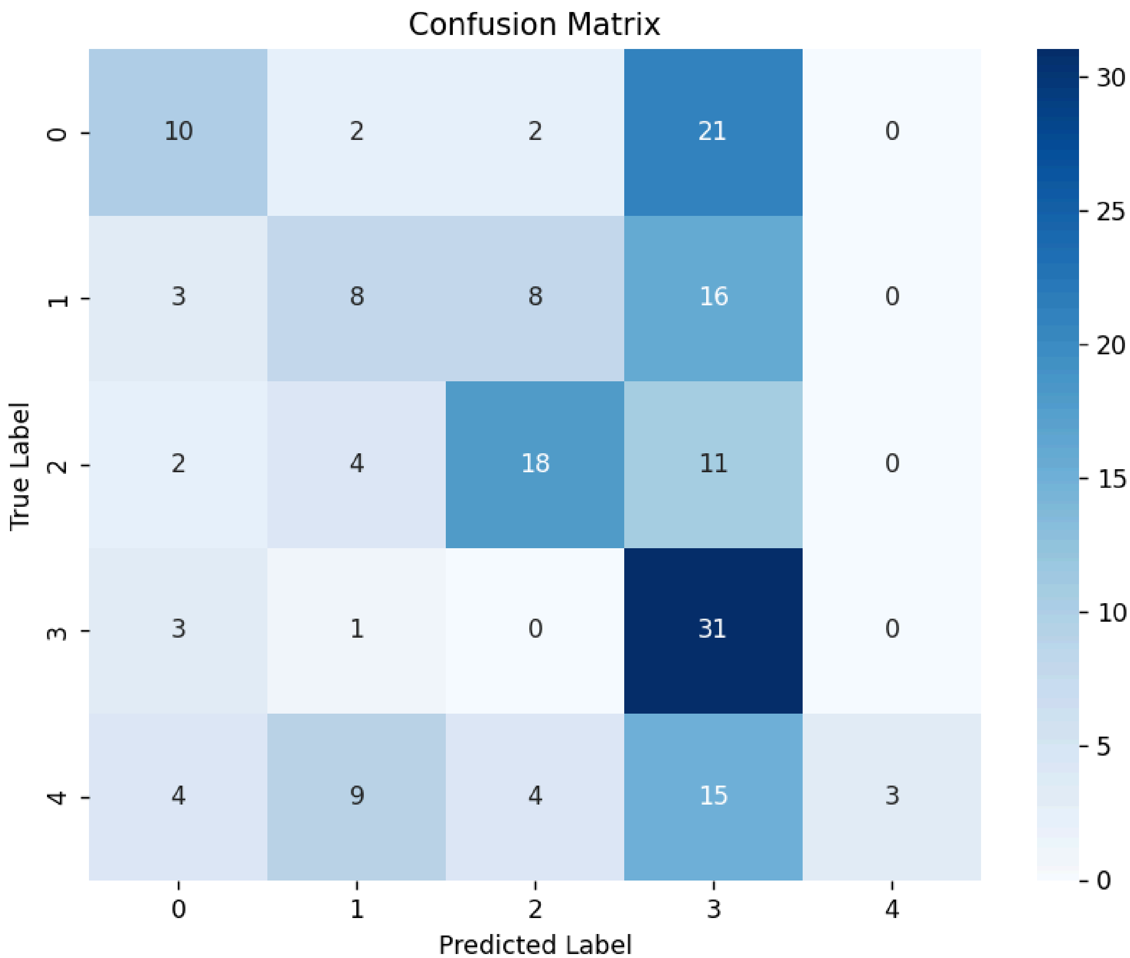
In hw3 we trained a model that predicts the sentiment of a sentence with words only from 1 list(Israeli **or** Palestinian word) . In this part we want to check the performance of the model with sentences that contain words from **both** lists. And then we Evaluated the model using precision, recall, F1 score, and accuracy.

From the file posts_first_targil we ran the same code from hw3 with a line difference, now we want the sentences with words from both lists (palestinian words and israeli words)

By hand we extracted 35 sentences from each of the 5 categories:

```
label_mapping = {
    "anti-i": 0,
    "pro-p": 1,
    "neutral": 2,
    "anti-p": 3,
    "pro-i": 4
}
```

Results:



It seems like the model does not work so well for sentences with both types of words. It has a strong bias towards anti palestinian. It makes sense because the dataset is from the news of a war time. And most of the world thinks that Israel is the bad guy here and that they are hurting the Palestinians for no reason.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.29      0.35        35
           1       0.33      0.23      0.27        35
           2       0.56      0.51      0.54        35
           3       0.33      0.89      0.48        35
           4       1.00      0.09      0.16        35

    accuracy                           0.40       175
   macro avg       0.54      0.40      0.36       175
weighted avg       0.54      0.40      0.36       175
```

Example of a sentence that can be labeled anti israeli(by hand), and anti palestine(model):
gaza's health ministry says nearly 31,000 people - mostly women and children - have been killed by israel's offensive.
If we did a model of pro-Israeli anti-israel and neutral it would work better because it looks like it is confusing the anti palestinian and pro palestine. Or pro Israel with anti palestine. So if it was 3 categories it would work better.

The output file:

	A	B	C	D	E	F	G
1	Newspape	ument Nur	tence Nun	Sentence	Label By Hand	Prediction	Score
2	al-j	217	1	tel aviv/west jerusalem –	4	4	0.526748
3	jpost	524	4	In response, the IDF stru	4	0	0.6215764
4	jpost	416	3	If a meeting between Abl	2	2	0.9358292
5	jpost	558	2	The Israeli military dropp	3	3	0.8538795
6	BBC	223	1	an israeli missile has hit	4	0	0.6565936
7	The New Y	2	5	the israeli military also sa	3	3	0.5182915
8	jpost	548	5	Hezbollah has linked wh	1	3	0.9733946
9	al-j	37	2	israel is using bureaucra	0	0	0.6667844
10	jpost	519	6	He blamed Hamas for th	3	3	0.9763172
11	BBC	49	1	"the war in gaza crushed	1	3	0.9635468
12	BBC	23	2	it comes a day after the	3	3	0.9758857
13	The New Y	22	1	iran seizes commercial s	1	3	0.944811
14	BBC	42	3	the welsh singer, who ha	1	1	0.9508616
15	al-j	29	1	this year, the shadow of	1	3	0.9506314

Part B:

We create 6 models. SVM, LoR and ANN. on bert and then on Sbert embedding.

First we embed the sentences from hw4.(the ones with words from one list) in hw4 we put in labels.
We used models:
Bert: bert-based-uncase
Sbert: `all-MiniLM-L6-v2`

For the Bert Model we tokenized the sentences, removed stop words and then summed up the vectors (without the stop words) to create one vector for each document.
Suggestions For Improvement:

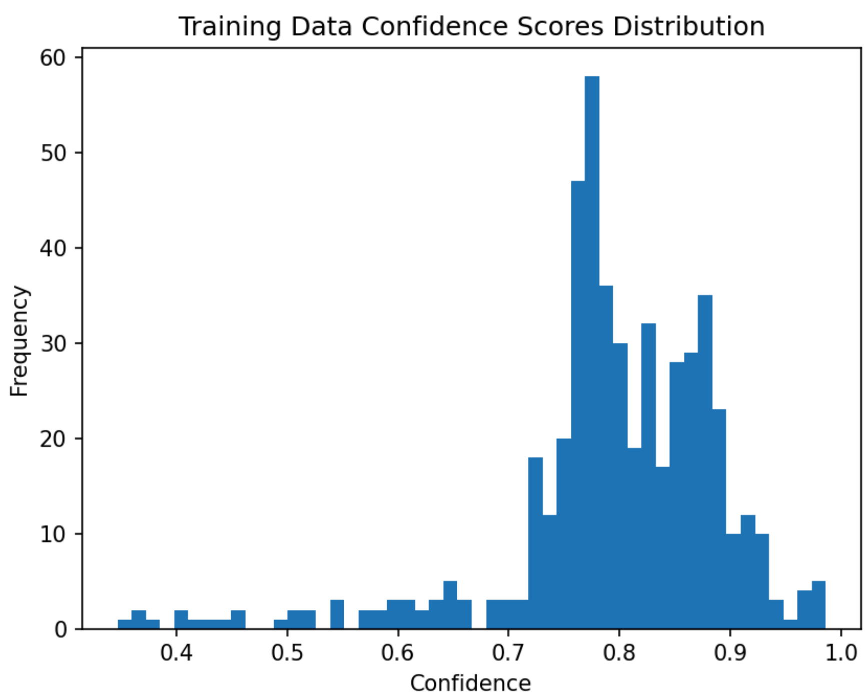
Removing stop words is a good start, but there are many other words that can be removed from our sentences before they are turned into vectors. Some words are expected with high probability to appear in proximity to each other. For example, if a document contains the word “starvation” we might expect to see the words: “strategy”, “policy”, “of [children, people, etc]” “as a weapon of war”, etc. These words do not help infer the meaning of the sentence and only distract (since we assume starvation is saying something negative about the people utilizing it) therefore those words can be removed. We could have a dictionary of words that have the connotation of war, and keep only those words.

Since we are dealing with sentences then we can assume that all of the sentences have less than 512 tokens/words. That's why we didnt take care of the limitation that Bert works with up to 512 tokens.

Support Vector Machine (SVM):

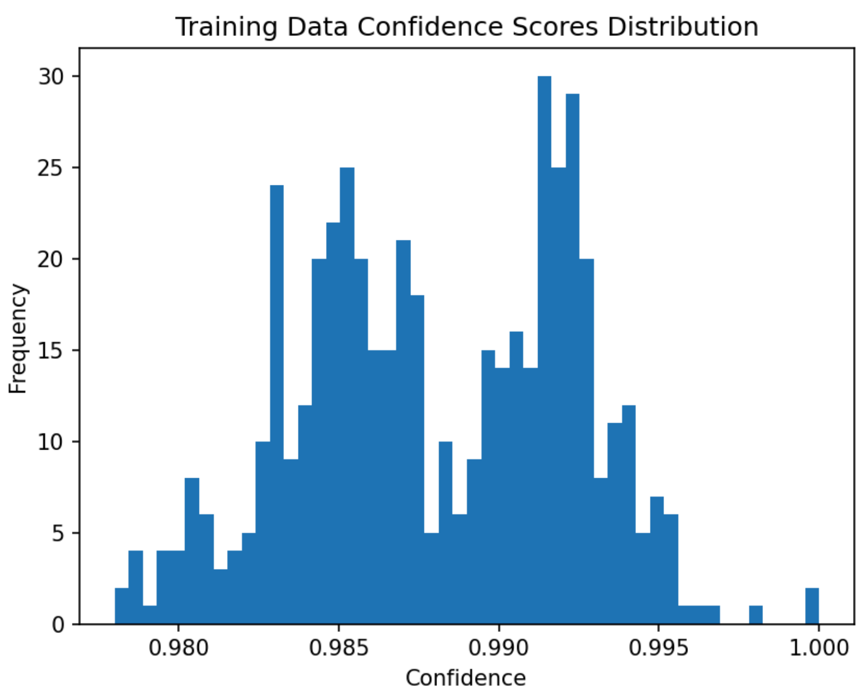
Perform 10-fold cross-validation to evaluate the model.
-Bert

```
Best Parameters: {'C': 10, 'gamma': 'scale'}
Accuracy: 0.5920
Precision: 0.6511
Recall: 0.5920
F1 Score: 0.5955
Confusion Matrix:
[[10  0  3  7  4]
 [ 1 22  4  1  9]
 [ 0  0  8  2  9]
 [ 3  4  2 18  0]
 [ 0  0  2  0 16]]
```



-Sbert

```
Best Parameters: {'C': 10, 'gamma': 'scale'}
Accuracy: 0.7200
Precision: 0.7531
Recall: 0.7200
F1 Score: 0.7214
Confusion Matrix:
[[17  0  5  2  0]
 [ 0 24  3  3  7]
 [ 2  0  9  3  5]
 [ 1  1  2 23  0]
 [ 0  1  0  0 17]]
```



Logistic Regression (LoR):

Perform 10-fold cross-validation to evaluate the model.
-Bert

```
Cross-Validation Accuracy: 0.6400
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.66      0.66       125
           1       0.64      0.70      0.67       125
           2       0.58      0.55      0.57       125
           3       0.68      0.69      0.68       125
           4       0.63      0.59      0.61       125

    accuracy                           0.64       625
   macro avg       0.64      0.64      0.64       625
weighted avg       0.64      0.64      0.64       625

Predictions saved to BERT_file.xlsx
```

-Sbert

```
Cross-Validation Accuracy: 0.7392
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.70      0.71       125
           1       0.79      0.76      0.78       125
           2       0.69      0.65      0.67       125
           3       0.75      0.77      0.76       125
           4       0.74      0.82      0.78       125

    accuracy                           0.74       625
   macro avg       0.74      0.74      0.74       625
weighted avg       0.74      0.74      0.74       625
```

Artificial Neural Network (ANN):

ANN model with 4 hidden layers
-Bert

```
Epoch 1/15
14/16 [=========================>....] - ETA: 0s - loss: 3.2041 - accuracy: 0.2299
Epoch 1: val_accuracy improved from -inf to 0.29464, saving model to best_model_bert_2.h5
C:\Users\user\AppData\Roaming\Python\Python38\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your mod
  saving_api.save_model(
16/16 [==============================] - 3s 63ms/step - loss: 3.0710 - accuracy: 0.2340 - val_loss: 1.9118 - val_accuracy: 0.2946
Epoch 2/15
11/16 [===================>..........] - ETA: 0s - loss: 1.6607 - accuracy: 0.3153
Epoch 2: val_accuracy did not improve from 0.29464
16/16 [==============================] - 0s 16ms/step - loss: 1.6235 - accuracy: 0.3140 - val_loss: 1.9959 - val_accuracy: 0.2946
Epoch 3/15
11/16 [===================>..........] - ETA: 0s - loss: 1.5070 - accuracy: 0.3665
Epoch 3: val_accuracy did not improve from 0.29464
16/16 [==============================] - 0s 10ms/step - loss: 1.4456 - accuracy: 0.4020 - val_loss: 1.9976 - val_accuracy: 0.2768
Epoch 4/15
15/16 [===========================>..] - ETA: 0s - loss: 1.2944 - accuracy: 0.4563
Epoch 4: val_accuracy improved from 0.29464 to 0.34821, saving model to best_model_bert_2.h5
16/16 [==============================] - 0s 13ms/step - loss: 1.2948 - accuracy: 0.4560 - val_loss: 1.8982 - val_accuracy: 0.3482
Epoch 5/15
14/16 [=========================>....] - ETA: 0s - loss: 1.2561 - accuracy: 0.4821
Epoch 5: val_accuracy did not improve from 0.34821
16/16 [==============================] - 0s 9ms/step - loss: 1.2724 - accuracy: 0.4700 - val_loss: 1.8104 - val_accuracy: 0.3393
Epoch 6/15
14/16 [=========================>....] - ETA: 0s - loss: 1.1996 - accuracy: 0.5335
```



```
Epoch 6: val_accuracy improved from 0.34821 to 0.36607, saving model to best_model_bert_2.h5
16/16 [==============================] - 0s 13ms/step - loss: 1.1797 - accuracy: 0.5380 - val_loss: 1.7270 - val_accuracy: 0.3661
Epoch 7/15
13/16 [=======================>......] - ETA: 0s - loss: 1.1007 - accuracy: 0.6034
Epoch 7: val_accuracy improved from 0.36607 to 0.41071, saving model to best_model_bert_2.h5
16/16 [==============================] - 0s 15ms/step - loss: 1.1175 - accuracy: 0.5800 - val_loss: 1.9347 - val_accuracy: 0.4107
Epoch 8/15
14/16 [=========================>....] - ETA: 0s - loss: 1.0197 - accuracy: 0.6362
Epoch 8: val_accuracy did not improve from 0.41071
16/16 [==============================] - 0s 10ms/step - loss: 1.0272 - accuracy: 0.6380 - val_loss: 1.9651 - val_accuracy: 0.3750
Epoch 9/15
14/16 [=========================>....] - ETA: 0s - loss: 0.9747 - accuracy: 0.6384
Epoch 9: val_accuracy did not improve from 0.41071
16/16 [==============================] - 0s 9ms/step - loss: 0.9783 - accuracy: 0.6400 - val_loss: 2.1654 - val_accuracy: 0.3750
Epoch 10/15
12/16 [=====================>........] - ETA: 0s - loss: 0.8977 - accuracy: 0.6823
Epoch 10: val_accuracy did not improve from 0.41071
16/16 [==============================] - 0s 11ms/step - loss: 0.8852 - accuracy: 0.6860 - val_loss: 1.9157 - val_accuracy: 0.4107
Epoch 11/15
16/16 [==============================] - ETA: 0s - loss: 0.7691 - accuracy: 0.7440
Epoch 11: val_accuracy improved from 0.41071 to 0.43750, saving model to best_model_bert_2.h5
16/16 [==============================] - 0s 14ms/step - loss: 0.7691 - accuracy: 0.7440 - val_loss: 1.9182 - val_accuracy: 0.4375
Epoch 12/15
```



```
16/16 [==============================] - ETA: 0s - loss: 0.7022 - accuracy: 0.7560
Epoch 12: val_accuracy improved from 0.43750 to 0.45536, saving model to best_model_bert_2.h5
16/16 [==============================] - 0s 14ms/step - loss: 0.7022 - accuracy: 0.7560 - val_loss: 1.8958 - val_accuracy: 0.4554
Epoch 13/15
 1/16 [>.............................] - ETA: 0s - loss: 0.6435 - accuracy: 0.7500
Epoch 13: val_accuracy did not improve from 0.45536
16/16 [==============================] - 0s 7ms/step - loss: 0.6360 - accuracy: 0.7880 - val_loss: 1.8538 - val_accuracy: 0.4196
Epoch 14/15
15/16 [===========================>..] - ETA: 0s - loss: 0.5870 - accuracy: 0.8083
Epoch 14: val_accuracy improved from 0.45536 to 0.49107, saving model to best_model_bert_2.h5
16/16 [==============================] - 0s 12ms/step - loss: 0.5877 - accuracy: 0.8000 - val_loss: 1.8959 - val_accuracy: 0.4911
Epoch 15/15
16/16 [==============================] - ETA: 0s - loss: 0.5080 - accuracy: 0.8460
Epoch 15: val_accuracy did not improve from 0.49107
16/16 [==============================] - 0s 8ms/step - loss: 0.5080 - accuracy: 0.8460 - val_loss: 1.7958 - val_accuracy: 0.4911
Best Epoch Used: 14
4/4 [==============================] - 0s 3ms/step
1/1 [==============================] - 0s 44ms/step - loss: 1.4133 - accuracy: 0.6154
Test Accuracy: 0.6153846383094788
```

-Sbert

```
Epoch 1/15
 1/16 [>.............................] - ETA: 31s - loss: 1.6093 - accuracy: 0.1875
C:\Users\user\AppData\Roaming\Python\Python38\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your model
  saving_api.save_model(
Epoch 1: val_accuracy improved from -inf to 0.26786, saving model to best_model_sbert_2.h5
16/16 [==============================] - 3s 35ms/step - loss: 1.6063 - accuracy: 0.2600 - val_loss: 1.5997 - val_accuracy: 0.2679
Epoch 2/15
 1/16 [>.............................] - ETA: 0s - loss: 1.5955 - accuracy: 0.3125
Epoch 2: val_accuracy improved from 0.26786 to 0.29464, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 10ms/step - loss: 1.5801 - accuracy: 0.3300 - val_loss: 1.5592 - val_accuracy: 0.2946
Epoch 3/15
 1/16 [>.............................] - ETA: 0s - loss: 1.5274 - accuracy: 0.3750
Epoch 3: val_accuracy improved from 0.29464 to 0.32143, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 9ms/step - loss: 1.5137 - accuracy: 0.3820 - val_loss: 1.4811 - val_accuracy: 0.3214
Epoch 4/15
 1/16 [>.............................] - ETA: 0s - loss: 1.4612 - accuracy: 0.4375
Epoch 4: val_accuracy improved from 0.32143 to 0.33036, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 10ms/step - loss: 1.3992 - accuracy: 0.4020 - val_loss: 1.4000 - val_accuracy: 0.3304
Epoch 5/15
 1/16 [>.............................] - ETA: 0s - loss: 1.2759 - accuracy: 0.4375
Epoch 5: val_accuracy improved from 0.33036 to 0.38393, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 9ms/step - loss: 1.2574 - accuracy: 0.4420 - val_loss: 1.3099 - val_accuracy: 0.3839
Epoch 6/15
```

```
 1/16 [>.............................] - ETA: 0s - loss: 1.2574 - accuracy: 0.5000
Epoch 6: val_accuracy improved from 0.38393 to 0.48214, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 10ms/step - loss: 1.1003 - accuracy: 0.5340 - val_loss: 1.2054 - val_accuracy: 0.4821
Epoch 7/15
 1/16 [>.............................] - ETA: 0s - loss: 0.9903 - accuracy: 0.6562
Epoch 7: val_accuracy improved from 0.48214 to 0.60714, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 12ms/step - loss: 0.9246 - accuracy: 0.6920 - val_loss: 1.0672 - val_accuracy: 0.6071
Epoch 8/15
 1/16 [>.............................] - ETA: 0s - loss: 0.7464 - accuracy: 0.7812
Epoch 8: val_accuracy improved from 0.60714 to 0.63393, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 12ms/step - loss: 0.7467 - accuracy: 0.7940 - val_loss: 0.9830 - val_accuracy: 0.6339
Epoch 9/15
 1/16 [>.............................] - ETA: 0s - loss: 0.6246 - accuracy: 0.8750
Epoch 9: val_accuracy improved from 0.63393 to 0.64286, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 11ms/step - loss: 0.5925 - accuracy: 0.8440 - val_loss: 0.9140 - val_accuracy: 0.6429
Epoch 10/15
 1/16 [>.............................] - ETA: 0s - loss: 0.4388 - accuracy: 0.8438
Epoch 10: val_accuracy did not improve from 0.64286
16/16 [==============================] - 0s 6ms/step - loss: 0.4703 - accuracy: 0.8700 - val_loss: 0.9330 - val_accuracy: 0.6429
Epoch 11/15
 1/16 [>.............................] - ETA: 0s - loss: 0.5322 - accuracy: 0.7188
Epoch 11: val_accuracy improved from 0.64286 to 0.66071, saving model to best_model_sbert_2.h5
16/16 [==============================] - 0s 11ms/step - loss: 0.3784 - accuracy: 0.8980 - val_loss: 0.9061 - val_accuracy: 0.6607
Epoch 12/15
```

```
 1/16 [>............................] - ETA: 0s - loss: 0.3662 - accuracy: 0.9375
Epoch 12: val_accuracy did not improve from 0.66071
16/16 [==============================] - 0s 6ms/step - loss: 0.3078 - accuracy: 0.9220 - val_loss: 0.8735 - val_accuracy: 0.6607
Epoch 13/15
 1/16 [>............................] - ETA: 0s - loss: 0.3055 - accuracy: 0.9688
Epoch 13: val_accuracy did not improve from 0.66071
16/16 [==============================] - 0s 7ms/step - loss: 0.2528 - accuracy: 0.9400 - val_loss: 0.9270 - val_accuracy: 0.6429
Epoch 14/15
 1/16 [>............................] - ETA: 0s - loss: 0.1570 - accuracy: 1.0000
Epoch 14: val_accuracy did not improve from 0.66071
16/16 [==============================] - 0s 6ms/step - loss: 0.2100 - accuracy: 0.9520 - val_loss: 0.9425 - val_accuracy: 0.6607
Epoch 15/15
 1/16 [>............................] - ETA: 0s - loss: 0.1469 - accuracy: 1.0000
Epoch 15: val_accuracy did not improve from 0.66071
16/16 [==============================] - 0s 6ms/step - loss: 0.1735 - accuracy: 0.9680 - val_loss: 0.9681 - val_accuracy: 0.6607
Best Epoch Used: 11
4/4 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 44ms/step - loss: 0.3887 - accuracy: 0.8462
Test Accuracy: 0.8461538553237915
```

Interesting insights:

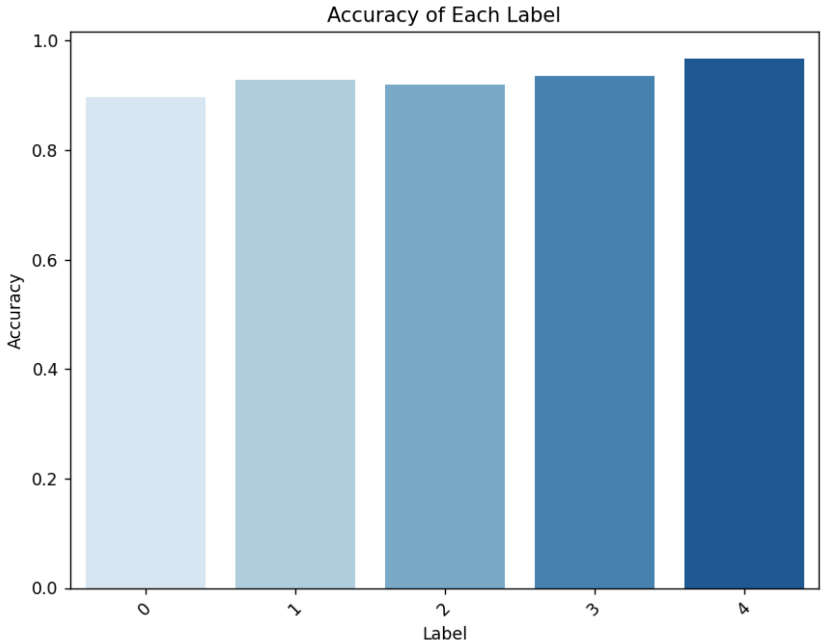
1) Within each model When the score was 1. Then the prediction was correct. All the time. The model was very sure of its prediction and it was really correct. Those sentences were very straight forward.
2) Because we are using 10 cross folds for SVM and LoR then it splits the data so sometimes each line is in the train and sometimes it's in the test. That's how it knows how to predict the label and score without knowing the data beforehand.
3) Ann testing was only on some of the dataset so only some sentences had predictions. So we had to merge the files smartly.

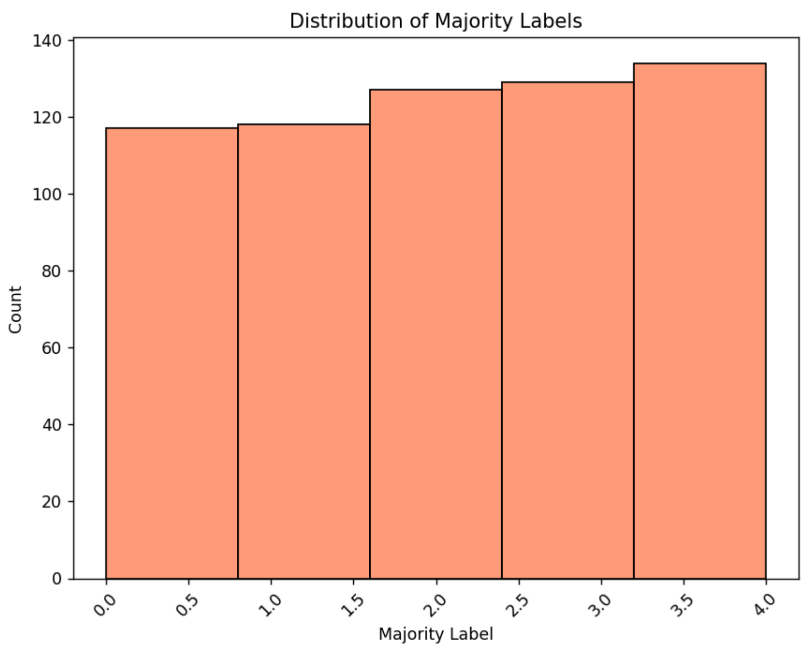
Majority:

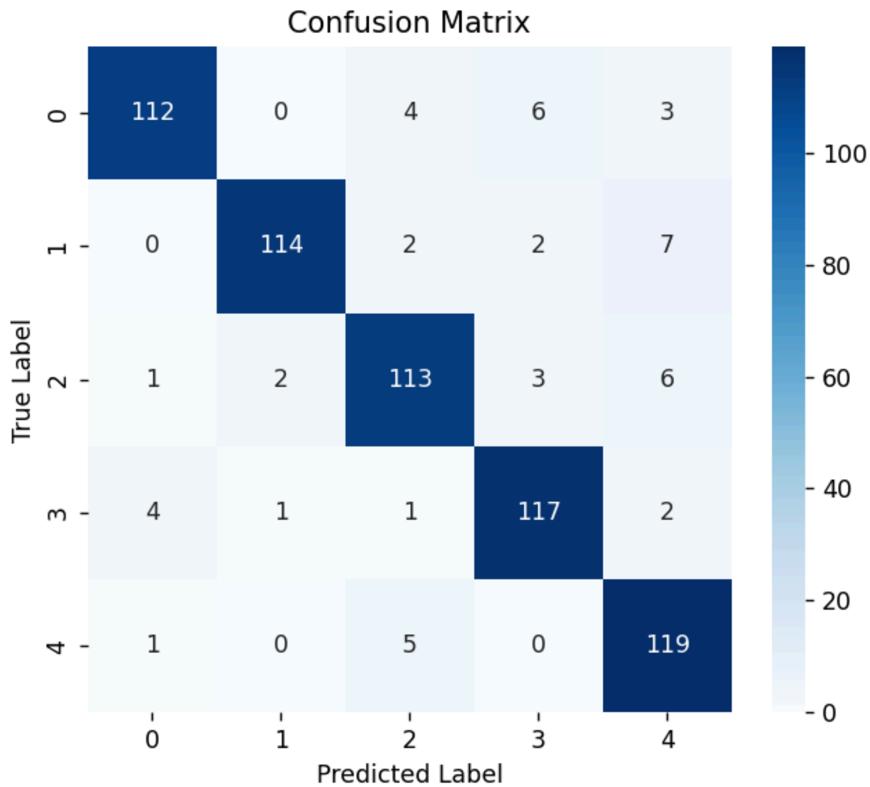
After we ran all models and combined them to 1 file we checked the majority label of the 6 models and if there was a tie we checked which average score was higher. We reached 92% accuracy (Accuracy: 0.9200). Which is very good. 🙂

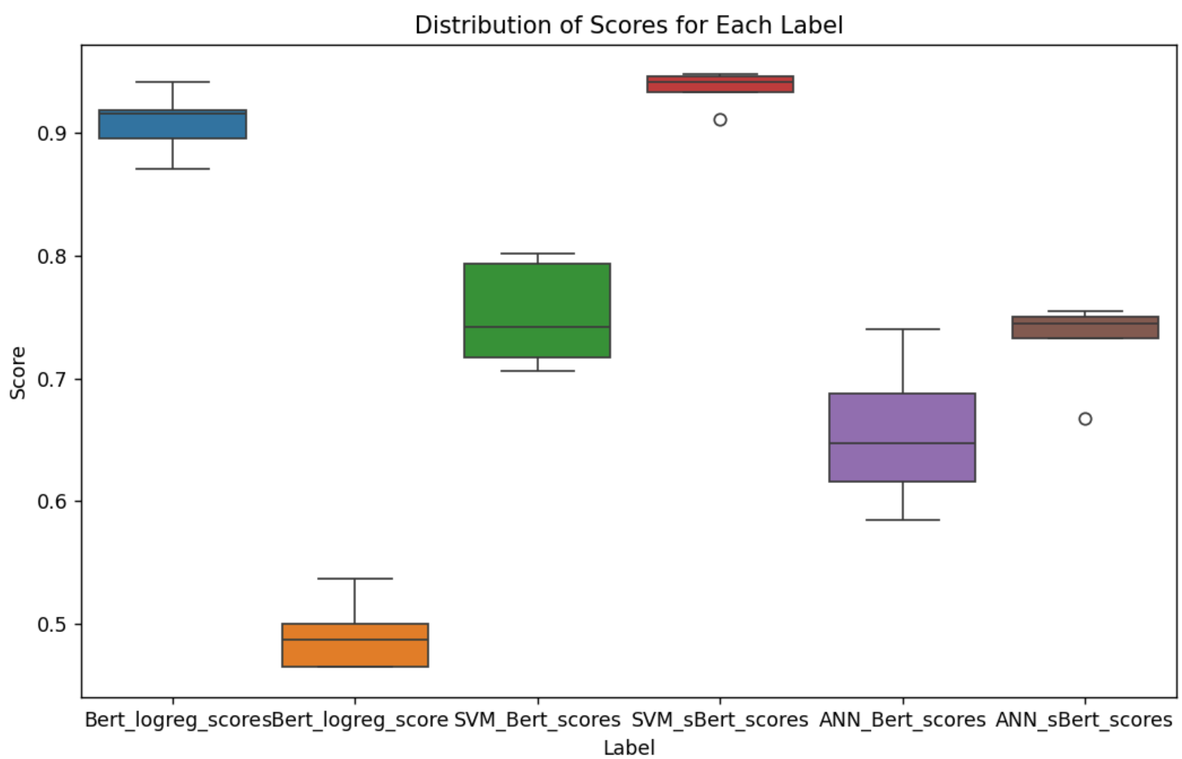
Output file:

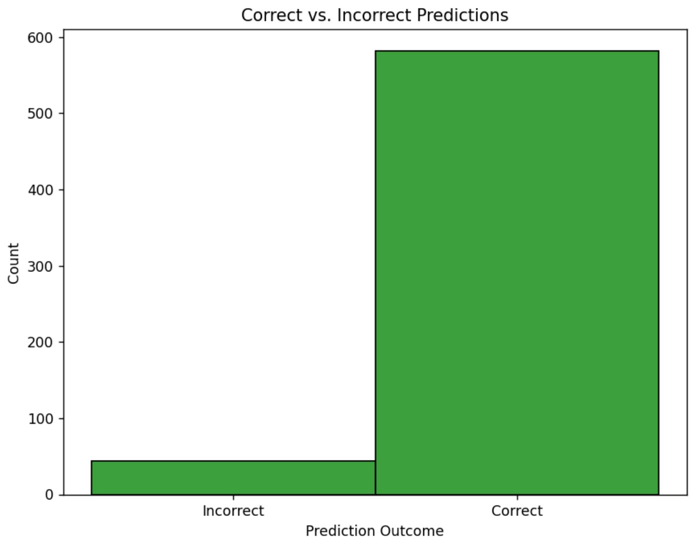
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Sentence	label	bel encode	edicted Lat	t_embedd	rt_embed	ogreg_pre	_logreg_s	ogreg_pre	t_logreg_s	Bert_pred	l_Bert_sc	Bert_pred	_sBert_sc	Bert_pred	l_Bert_sc	Bert_pred	_sBert_sc	ajority_Lat	ect_Prediction	
We aim to	pro-i	4	4	[-9.626074	[-0.058555	4	1	4	0.800774	4	0.759235	4	0.984286					4	1	
preparing t	pro-i	4	1	[-1.606126	[0.004436	1	0.84604	1	0.434483	4	0.799867	4	0.984836					4	1	
Museums i	pro-p	1	1	[-3.128663	[-0.041235	1	1	4	0.595729	4	0.489269	4	0.937045	4	0.838838	4	0.891639	4	0	
• Soci	pro-i	4	4	[-1.123837	[0.106906	1	0.521777	4	0.577329	4	0.883172	4	0.983091					4	1	
these inac	anti-i	0	0	[0.547733	[0.017690	0	1	0	0.57826	0	0.87677	0	0.986308					0	1	
IN F	pro-p	1	1	[-5.069488	[0.016631	1	0.974182	1	0.480772	1	0.790343	1	0.991718					1	1	

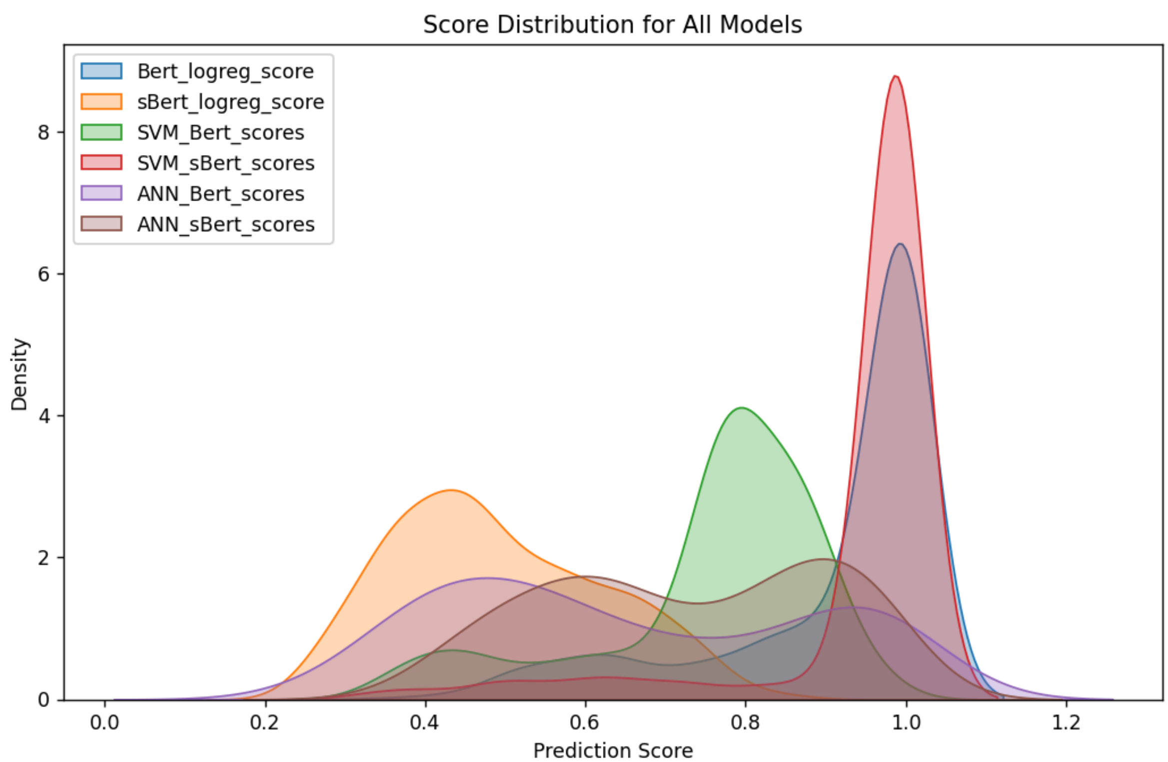

Accuracy of Each Label
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
Label


Distribution of Majority Labels
Count
140
120
100
80
60
40
20
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Majority Label


Confusion Matrix
True Label
Predicted Label
112
0
4
6
3
0
114
2
2
7
1
2
113
3
6
4
1
1
117
2
1
0
5
0
119
0
1
2
3
4
100
80
60
40
20


Distribution of Scores for Each Label
Score
0.9
0.8
0.7
0.6
0.5
Bert_logreg_scores
Bert_logreg_score
SVM_Bert_scores
SVM_sBert_scores
ANN_Bert_scores
ANN_sBert_scores
Label





Ideas for improvement:

One of the things we can do to improve our model is to use more accurate data. Our data is based on other models we built, and if we had labeled the data by hand we might be able to train it better.

Another thing we can do is play around with the hidden layers of the ANN model, and with the attributes of the other models to try to reach optimal results.