Assignment 3:

Team 10: Rama Krishna Kommineni, Rohith Nagabhyrava, Umair Akhtar

Experiment 1

1. **BOW model -**
   a) Using CountVectorizer and Multinomial Naive Bayes
   Acc - 0.537
   Confusion Matrix -
```
array([[ 50,  57,  18],
       [  3, 604,  58],
       [  2, 167, 355]], dtype=int64)
```
   b) Using tf-idf
   Acc - 0.553
   Confusion Matrix -
```
array([[  1,  96,  28],
       [  0, 632,  33],
       [  0, 214, 310]], dtype=int64)
```
   c) Keras tokenizer
   Acc - 0.468
   Confusion Matrix -
```
[[ 5 21  6]
 [32 82 52]
 [21 84 26]]
```

2. **GLOVE model_**
   a) **Using Pretrained glove vectors**
   Acc - 0.69
   Confusion Matrix-
```
[[ 33  14 115]
 [  9   3  27]
 [ 25   8 178]]
```

   b) **Tuning the model**
   Acc- 0.70
   Confusion Matrix-
```
[[ 80   2  80]
 [ 20   1  18]
 [ 59   6 146]]
```

Experiment 2

1. **BoW Model -**
   Accuracy - 0.19
   Confusion matrix
   ```
   [[  0 655]
    [  0 157]]
   ```
2. **GLOVE Model -**
   Accuracy - 0.81
   Confusion matrix
   ```
   [[165   3]
    [ 36   0]]
   ```

Experiment 3 and 4

Confusion Matrix from Google API

```
from sklearn.metrics import accuracy_score
print('accuracy is: ',accuracy_score(google.label,google.predict))
pd.crosstab(google.label,google.predict)
```

```
accuracy is:  0.4808743169398907
```

predict	negative	neutral	positive
label			
negative	13	65	79
neutral	52	360	421
positive	37	201	419

Confusion Matrix for Watson API

```
print('accuracy is: ',accuracy_score(google.label,ibm.predict) )
pd.crosstab(google.label,ibm.predict)
```

```
accuracy is:  0.49058894960534305
```

predict	negative	neutral	positive
label			
negative	15	119	23
neutral	40	634	159
positive	35	463	159

Confusion Matrix of Azure

```
print('accuracy is: ',accuracy_score(google.label,azure.sentiment_predicted) )
pd.crosstab(google.label,azure.sentiment_predicted)
```

```
accuracy is:  0.44019429265330906
```

sentiment_predicted	negative	neutral	positive
label			
negative	14	90	53
neutral	42	437	354
positive	47	336	274

Average Accuracy of All API

```
print('accuracy of all API average is :', accuracy_score(mode['mode1'],google['label'].replace({'positive'
:0,'negative':1,'neutral':2})))
pd.crosstab(mode['mode1'],google['label'].replace({'positive':0,'negative':1,'neutral':2}))
```

```
accuracy of all API average is : 0.4820886460230723
```

label	0	1	2
mode1			
0.0	316	59	348
1.0	15	4	11
2.0	326	94	474

Auto ML:

Best Model from TPOT:

```
TPOT closed prematurely. Will use the current best pipeline.

Best pipeline: XGBClassifier(LinearSVC(LinearSVC(input_matrix, C=0.1, dual=False, loss=squared_hinge, pena
lty=l2, tol=0.1), C=0.1, dual=False, loss=squared_hinge, penalty=l1, tol=0.1), learning_rate=0.1, max_dept
h=1, min_child_weight=13, n_estimators=100, nthread=1, subsample=0.6500000000000001)
0.0
```

Best Model from H20.ai: H20 gave us the best accuracy as shown below

```
H2O auto ML accuracy score is 0.5726495726495726
```

```
              precision    recall  f1-score   support

         0.0       0.50      0.57      0.53        83
         1.0       0.00      0.00      0.00         0
         2.0       0.76      0.58      0.65       151

   micro avg       0.57      0.57      0.57       234
   macro avg       0.42      0.38      0.40       234
weighted avg       0.67      0.57      0.61       234
```

label	0	1	2
predict			
0.0	47	8	28
2.0	47	17	87

GE Test:

Our best model was found using glove model during transfer learning on IMDB data set. When the model was used to predict GE dataset

```
[ ]  score = model1.evaluate(X_padGE, yGE, verbose=1)
     print('Test Loss:', score[0], 'Test Accuracy:', score[1])
```

```
131/131 [==============================] - 0s 65us/step
Test Loss: 0.47921781913014766 Test Accuracy: 0.8778625958748446
```

```
[ ]  from sklearn.metrics import confusion_matrix
     y_pred = model1.predict(X_padGE)
     print(confusion_matrix(yGE.argmax(axis=1), y_pred.argmax(axis=1)))
```

```
[[115   0]
 [ 16   0]]
```

Discuss what you learned from this exercise?

Learned how to perform sentiment analysis, fine tuning and ensemble learning from models that we build and from major cloud platforms.