CSYE 7374 : SPECIAL TOPICS IN COMPUTER SYSTEMS ENGINEERING

ASSIGNMENT 3

SENTIMENTAL ANALYSIS ON FINANCIAL DATASET

Dharani Thirumalaisamy

Manasa Bhimaraya Singhekar

INDEX

EXPERIMENT1:

1. Mix all the paragraphs and split the data to a 80-20 split

2. Build the 3 models listed earlier and compute the confusion matrix for the training and testing datasets.

3. Discuss how your models perform

Solution: Preview of the data

Out[7]:

	text	Sentiments
0	Netflix, Inc. (NASDAQ:NFLX) Q3 2018 Earnings C...	Neutral
1	Executives	Neutral
10	Good afternoon and welcome to the Netflix Q3 2...	Positive
100	So a couple questions; number one, what have y...	Positive
101	Reed Hastings	Neutral
102	Ted, do you want to take that?	Neutral
103	Ted Sarandos	Neutral
104	Yes, I would say that, one thing that we've le...	Positive
105	So what we're learning more and more is that, ...	Positive
106	So that gives us – and we do it over many titl...	Positive
107	Spencer Wang	Neutral
108	And Eric just sort of on the financial aspect ...	Negative
109	Eric Sheridan	Neutral
11	As a reminder, we will be making forward-looki...	Neutral
110	And that would lead to maybe my next question,...	Positive
111	David Wells	Neutral
112	It's either for Ted or I'll take that the sort...	Positive
113	Ted Sarandos	Neutral
114	Yes. We've really been trying to optimize and ...	Positive
115	Eric Sheridan	Neutral
116	Following up on that, the comment from the let...	Neutral
117	David Wells	Neutral
118	It's mostly the latter Eric, so we're not acti...	Negative
119	Eric Sheridan	Neutral
12	With that, over to you now Eric, for the first...	Neutral

Bag of Words:

Pre-processing steps

1.Tokenization .

2.All common separators, operators, **punctuations** and non-printable characters are removed .

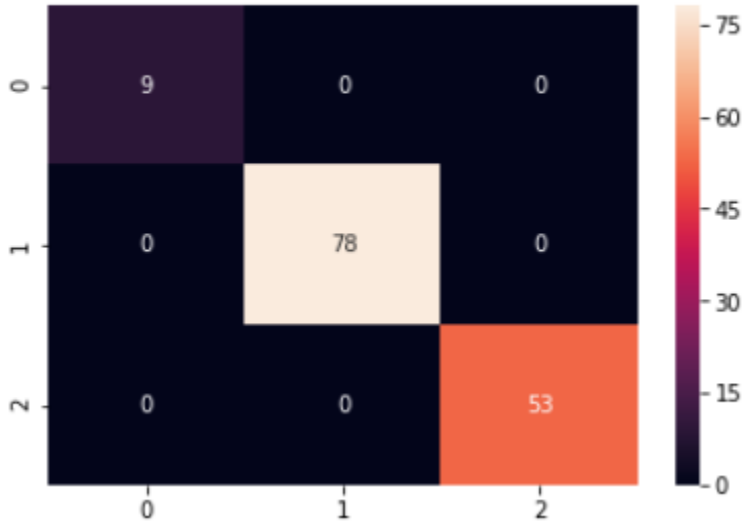
3.Then, **stop-words** are filtered. Samples: aba but, maybe, wonder "- wonder but, maybe, I wonder S.

4.Finally, the stemming and / or lemmatization is applied to the stem. In the article, we skip the stemming.

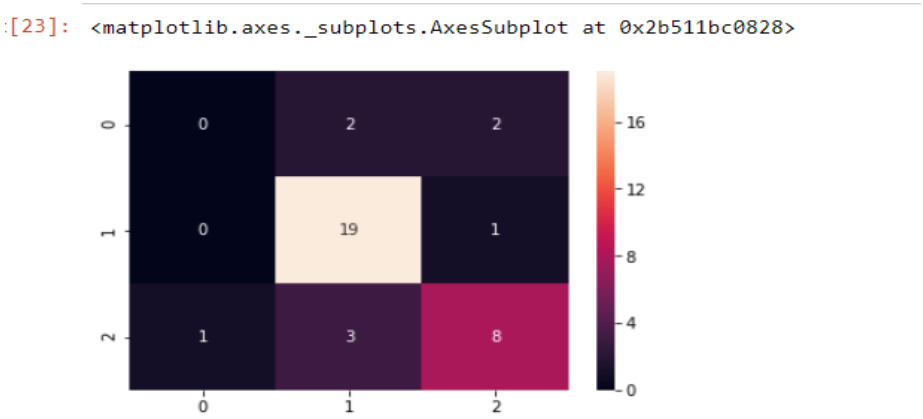
Model accuracy on test data is **0.75**.

Confusion Matrix: Train

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x2b511c54550>
```



Test data:



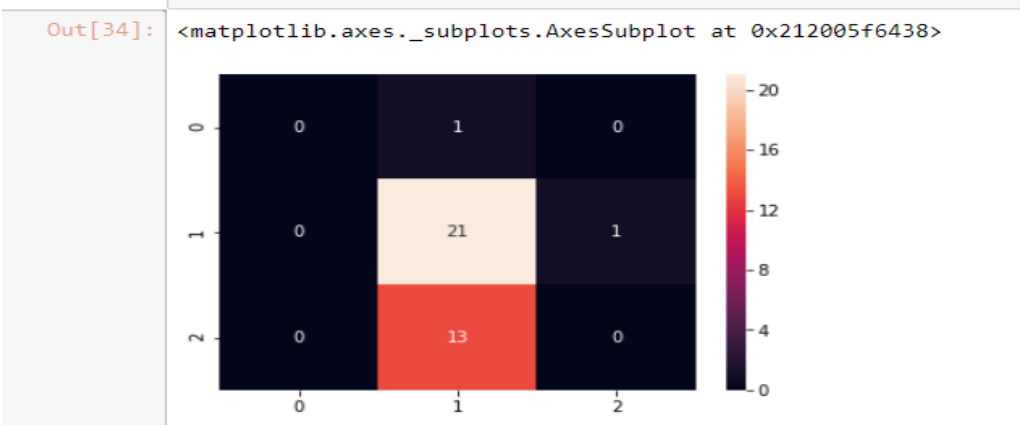
Test prediction:

Neutral- 19 out of 24 labels were correct.

Positive:8/12

Negative:0/1

GloVe



The accuracy of the model was low for any kind of correct prediction

RNN_Model

Test predictions:

```
array([[ 0,  0,  4],
       [ 0, 16,  6],
       [ 0,  0, 10]], dtype=int64)
```

Neutral: 16/16

Positive:10/20

EXPERIMENT2:

BagOfWords :

```
7]: from sklearn import metrics
    cnf_matrix = metrics.confusion_matrix(y_test, predictions)
    cnf_matrix

7]: array([[10758,  1742],
           [ 1749, 10751]], dtype=int64)

9]: y_train_predictions=clf.predict(x_train)
    y_train_predictions

9]: array([1., 0., 0., ..., 0., 1., 0.], dtype=float32)

0]: from sklearn import metrics
    cnf_matrix = metrics.confusion_matrix(y_train, y_train_predictions)
    cnf_matrix

0]: array([[11798,   702],
           [  635, 11865]], dtype=int64)

1]: scores=clf.score(x_train,y_train)
    scores

1]: 0.94652
```

GloVe model:

Network configuration:

```
: from keras.models import Sequential
  from keras.layers import Embedding, Flatten, Dense, Input, LSTM, Embedding, Dropout, Activation,Bidirectional

  model = Sequential()
  model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
  model.add((LSTM(50, return_sequences=True, dropout=0.5)))
  model.add(Flatten())
  model.add(Dense(32, activation='relu'))



  model.add(Dense(1, activation='sigmoid'))
  #model.summary()

  model.layers[0].set_weights([embedding_matrix])
  model.layers[0].trainable = False
  model.summary()

  model.compile(optimizer='sgd',
                loss='binary_crossentropy',
                metrics=['acc'])
  history = model.fit(x_train, y_train,
                      epochs=80,
                      batch_size=512,
                      validation_data=(x_val, y_val))
  model.save_weights('pre_trained_glove_model.h5')
```

```
score=model.evaluate(data_t,labels_test)
```

```
25000/25000 [==============================] - 13s 507us/step
```

```
print('Test accuracy:',score[1])
```

```
Test accuracy 0.55064
```

```
scores_train=model.evaluate(x_train,y_train)
print('Train accuracy:',scores_train[1])
```

```
10000/10000 [==============================] - 5s 487us/step
Train accuracy: 0.6846
```

```
#Confusion Matrix train data
```

```
train_pred=model.predict(x_train)
```

```
train_y=np.int8(y_train)
train_cm=np.int8(train_pred.round())
```

```
cnf_matrix_train = metrics.confusion_matrix(train_y ,train_cm)
cnf_matrix_train
```

```
array([[3750, 1225],
       [1929, 3096]], dtype=int64)
```

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(test_y ,test_cm)
cnf_matrix
```

```
array([[7331, 5169],
       [6065, 6435]], dtype=int64)
```

The above confusion matrix for imdb indicates

Positive sentiment:7331/13,396

Negative sentiment: 6435/11,604

RNN_IMDB:

Network configuration:

```
from keras import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
embedding_size=32
model=Sequential()
model.add(Embedding(vocabulary_size, embedding_size, input_length=max_words))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
print(model.summary())
```

```
#Train
predictions_train=model.predict(X_train2)
```

```
y_train_p=np.int8(y_train2)
p_train2=np.int8(predictions_train.round())


cm_train=confusion_matrix(y_train_p, p_train2)
cm_train
```

```
array([[11803,   623],
       [  396, 12050]], dtype=int64)
```

```
#Test predictions
```

```
Test_predictions=model.predict(X_test)
```

```
scores=model.evaluate(X_test, y_test)
```

```
25000/25000 [==============================] - 130s 5ms/step
```

```
print('Test accuracy',scores[1])
```

```
Test accuracy 0.86564
```

```
test_y=np.int8(y_test)
test_pred=np.int8(Test_predictions.round())
cm_test=confusion_matrix(test_y, test_pred)
cm_test
```

```
array([[10634,  1866],
       [ 1493, 11007]], dtype=int64)
```

EXPERIMENT 3 :

Calculate the sentiment scores of the transcript using API's and compute the confusion matrix with manually labelled sentiments.

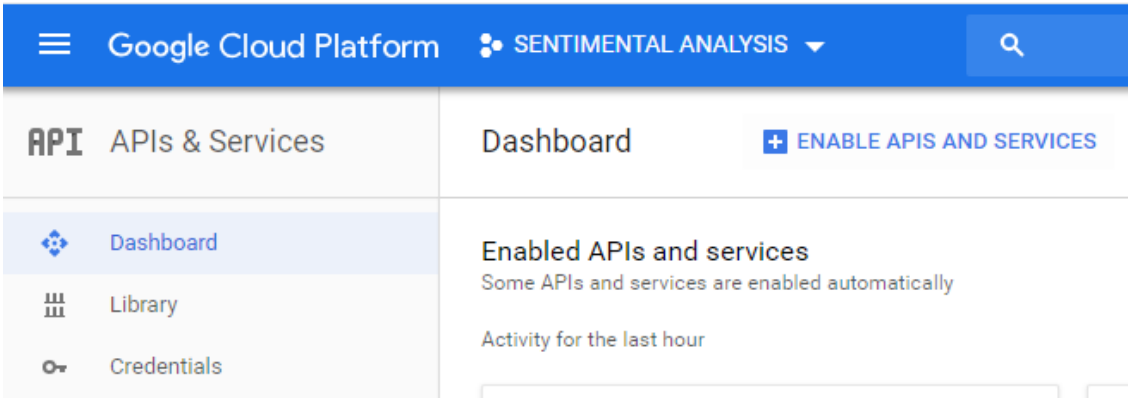
GOOGLE'S API :

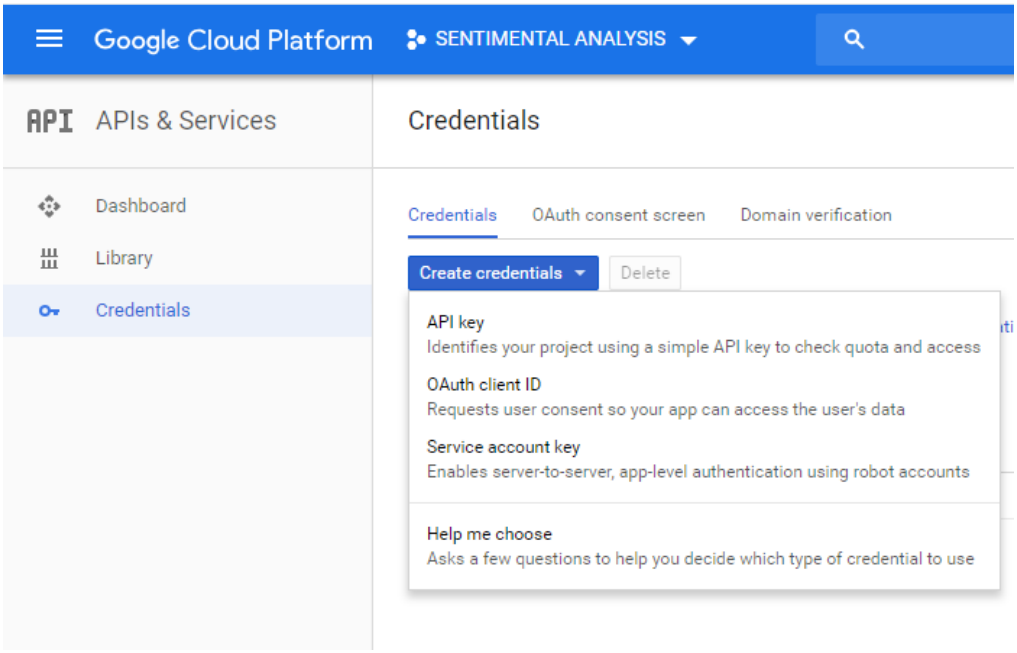
Google's Natural Language Text analytics gives a score ranging from -1 to 1.

Below zero indicates Negative sentiment, above 0 indicates positive sentiment and 0 is for Neutral sentiment.

	google_score
0	0.0
1	0.0
2	-0.2
3	0.0
4	-0.1
5	0.4
6	0.3
7	0.6
8	0.5
9	0.3

To connect to Google's API, Application Credential is required, which can be obtained once **API's are enabled and service account key is created.**





With the Application Credential, API's can be accessed from python script using this one line code.

```
client = language.LanguageServiceClient()
```

Before that, a json file will get downloaded to your system once you set up the credentials. That json file should be made as environment variable or it can be hardcoded.

```
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = your json file
```

Once the scores are obtained, the scores must be normalized to manual sentiment standard which is :

-1 – Negative ; 0 – Neutral ; 1 – Positive.

	google_score	Normalized Score
0	0.0	0.0
1	0.2	1.0
2	0.1	1.0
3	0.0	0.0
4	0.1	1.0
5	0.0	0.0
6	0.1	1.0
7	0.0	0.0
8	0.3	1.0
9	0.2	1.0

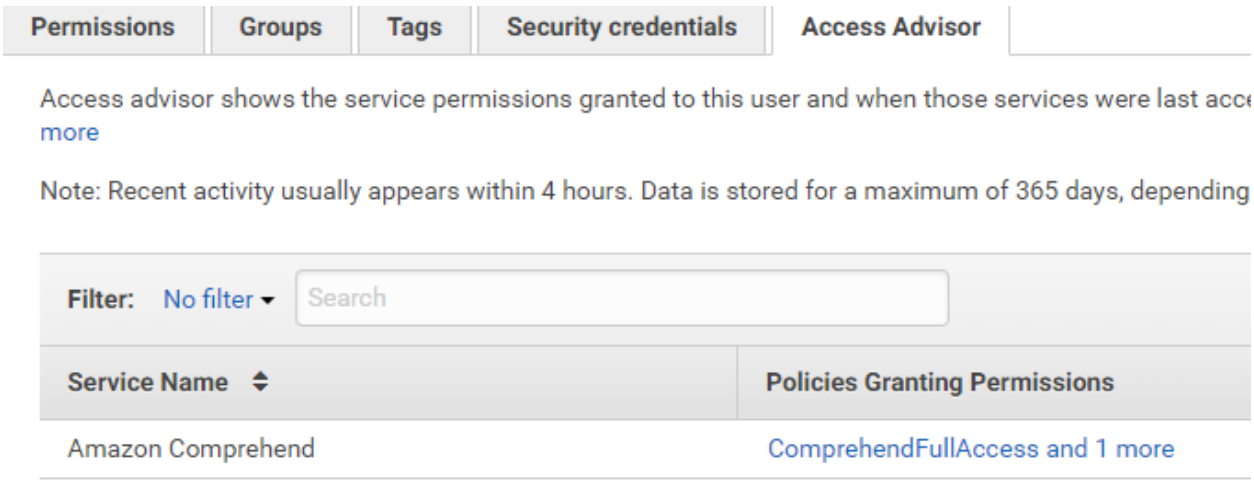
With this, Google score is ready.

AMAZON's API :

Similar to Google's API, scores must be calculated using Amazon's API too.

For this, first AWS account should be created. In that create An IAM user with full access to Amazon's Comprehend. Create Keys.

Access the API from the script using these keys.




```
comprehend = boto3.client(service_name='comprehend', region_name='us-east-1',aws_access_key_id=access_key,
    aws_secret_access_key=secret access key)
```

Using this code, Amazon's API can be connected from the script.

['{\n   "ResponseMetadata": {\n   "HTTPHeaders": {\n   "connection": "keep-alive",\n   "content-lengt
h": "166",\n   "content-type": "application/x-amz-json-1.1",\n   "date": "Mon, 26 Nov 2018 20:27:30 GMT",\n
            "x-amzn-requestid": "b2ecb096-f1b9-11e8-8656-5735cedac40a"\n   },\n   "HTTPStatusCode": 200,\n
"RequestId": "b2ecb096-f1b9-11e8-8656-5735cedac40a",\n   "RetryAttempts": 0\n   },\n   "Sentiment": "NEUTRAL",\n   "S
entimentScore": {\n   "Mixed": 0.00010634889622451738,\n   "Negative": 3.433611345826648e-05,\n   "Neutral":
0.9972013235092163,\n   "Positive": 0.002658019075170159 3\n   }\n}',

.This is how sentiment for each text is given. It has scores for Positive, Negative, Neutral, Mixed and overall sentiment too.

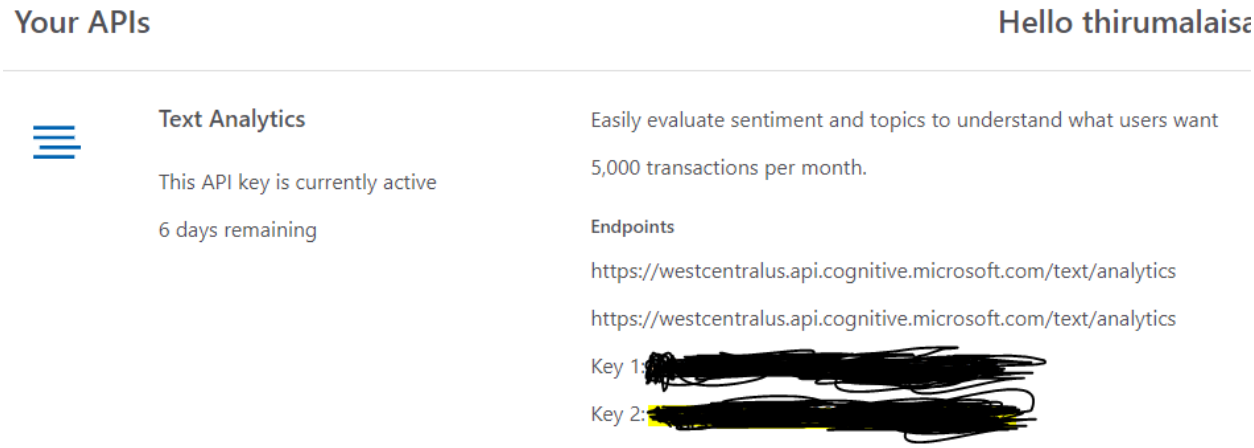
For this experiment, the overall score is considered and that is normalized to the manual sentiment standard.

	aws_score	Normalized aws score
0	NEUTRAL"	0
1	NEUTRAL"	0
2	NEUTRAL"	0
3	NEUTRAL"	0
4	NEUTRAL"	0
5	NEUTRAL"	0
6	NEUTRAL"	0
7	NEUTRAL"	0
8	POSITIVE	1
9	POSITIVE	1

MICROSOFT'S API:

Azure is Microsoft's cloud service.

Azure's text analytics API can be accesses for free. Similar to the above two API's even this one needs a subscription key to access the API from the script.



Any one key can be used to access the API.

```
headers = {'Ocp-Apim-Subscription-Key': your key}
conn = http.client.HTTPSConnection(url)
body = json.dumps (documents)
conn.request ("POST", path, body, headers)
response = conn.getresponse ()
return response.read ()
```


```
['{    "documents": [        {            "',
 '": "1",            "score": 0.5        },        {            "',
 '": "2",            "score": 0.8373502492904663        },        {            "',
 '": "3",            "score": 0.5        },        {            "',
 '": "4",            "score": 0.9589877128601074        },        {            "',
 '": "5",            "score": 0.5        },        {            "',
 '": "6",            "score": 0.7681329250335693        },        {            "',
 '": "7",            "score": 0.9077291488647461        },        {            "',
 '": "8",            "score": 0.21436676383018494        },        {            "',
 '": "9",            "score": 0.5        },        {            "',
 '": "10",            "score": 0.824548602104187        },        {            "',
 '": "11",            "score": 0.7773778438568115        },        {            "',
 '": "12",            "score": 0.11967316269874573        },        {            "',
 '": "13",            "score": 0.5        },        {            "',
 '": "14",            "score": 0.9447193741798401        },        {            "',
 '": "15",            "score": 0.5        },        {            "',
 '": "16",            "score": 0.875264048576355        },        {            "',
 '": "17",            "score": 0.5        },        {            "',
 '": "18",            "score": 0.16345813870429993        },        {            "',
```


In Azure , 0.5 is the threshold, below that is negative and above that it is positive.

So based on that, the scores are normalized.

	azure_score	Normalized azure score
0	0.5	0
1	0.837	1
2	0.5	0
3	0.958	1
4	0.5	0
5	0.768	1
6	0.907	1
7	0.214	-1
8	0.5	0
9	0.82	1

IBM's API :

Watson is IBM's cloud service. Watson's Natural Language Text analytics has a subscription key which has to be used with a unique URL given to you while enabling this API.

Service	Resource Group	Plan
Natural Language Understanding	Default	Lite

When you create this, API key and the URL will be shown. Use that to access the API.

```
service = NaturalLanguageUnderstandingV1(
     version='2018-03-16',
     url='https://gateway.watsonplatform.net/natural-language-understanding/api',
     iam_apikey=api)
```

There are many options available in Watson API, eg., Sentiment, emotions, Keywords, etc. Since in this experiment we need Sentiment, I used SentimentOptions.

```
[{'language': 'en',
  'sentiment': {'document': {'label': 'neutral', 'score': 0}},
  'usage': {'features': 1, 'text_characters': 86, 'text_units': 1}},
 {'language': 'en',
  'sentiment': {'document': {'label': 'positive', 'score': 0.743951}},
  'usage': {'features': 1, 'text_characters': 328, 'text_units': 1}},
 {'language': 'en',
  'sentiment': {'document': {'label': 'neutral', 'score': 0}},
  'usage': {'features': 1, 'text_characters': 148, 'text_units': 1}},
 {'language': 'en',
  'sentiment': {'document': {'label': 'positive', 'score': 0.759278}},
  'usage': {'features': 1, 'text_characters': 321, 'text_units': 1}},
 {'language': 'en',
  'sentiment': {'document': {'label': 'negative', 'score': -0.773955}},
  'usage': {'features': 1, 'text_characters': 463, 'text_units': 1}},
 {'language': 'en',
  'sentiment': {'document': {'label': 'neutral', 'score': 0}},
  'usage': {'features': 1, 'text_characters': 65, 'text_units': 1}},
```

The output will be in this format. Since this gives the overall sentiment, it can be directly normalized to manual sentiment standards.

	watson_score	Normalized watson score
0	neutral	0
1	positiv	1
2	neutral	0
3	positiv	1
4	negativ	-1
5	neutral	0
6	positiv	1
7	positiv	1
8	positiv	1
9	positiv	1

CONFUSION MATRIX :

To compute the confusion matrix between all the API's and manual sentiment, we must first decide on final sentiment for each text. To do so, maximum frequency for each row is calculated.

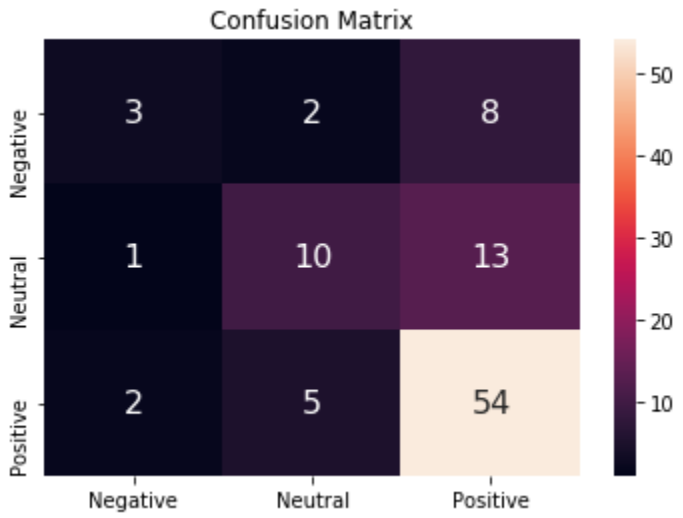
	google_score	Normalized Score	aws_score	Normalized aws score	azure_score	Normalized azure score	watson_score	Normalized watson score
0	0.0	0.0	NEUTRAL"	0	0.5	0	neutral	0
1	0.0	0.0	NEUTRAL"	0	0.5	0	positiv	1
2	-0.2	-1.0	NEUTRAL"	0	0.5	0	positiv	1
3	0.0	0.0	NEUTRAL"	0	0.5	0	neutral	0
4	-0.1	-1.0	POSITIVE	1	0.89	1	positiv	1
5	0.4	1.0	NEUTRAL"	0	0.87	1	positiv	1
6	0.3	1.0	NEGATIVE	-1	0.77	1	positiv	1
7	0.6	1.0	NEUTRAL"	0	0.5	0	neutral	0
8	0.5	1.0	NEUTRAL"	0	0.85	1	positiv	1
9	0.3	1.0	NEUTRAL"	0	0.9	1	positiv	1
10	0.8	1.0	NEUTRAL"	0	0.5	0	positiv	1
11	0.0	0.0	NEUTRAL"	0	0.5	0	positiv	1
12	0.3	1.0	NEUTRAL"	0	0.5	0	positiv	1

Calculating median for all the normalized rows :

	Highly Occuring Sentiment
0	0.0
1	1.0
2	0.0
3	1.0
4	0.0
5	0.0
6	1.0
7	0.0
8	1.0
9	1.0
10	1.0
11	-1.0

Calculating confusion matrix between this and the manual sentiments :

API SCORE VS NETFLIX DATASET :



PREDICTION ANALYSIS :

CORRECT PREDICTION :

NEGATIVE - 3 out of 13 ; NEUTRAL - 10 out of 24 ; POSITIVE - 54 out of 61

1) NEGATIVE LABEL ANALYSIS :

3 WERE PREDICTED AS NEGATIVE; 2 WERE PREDICTED AS NEUTRAL; 8 WERE PREDICTED AS POSITIVE.

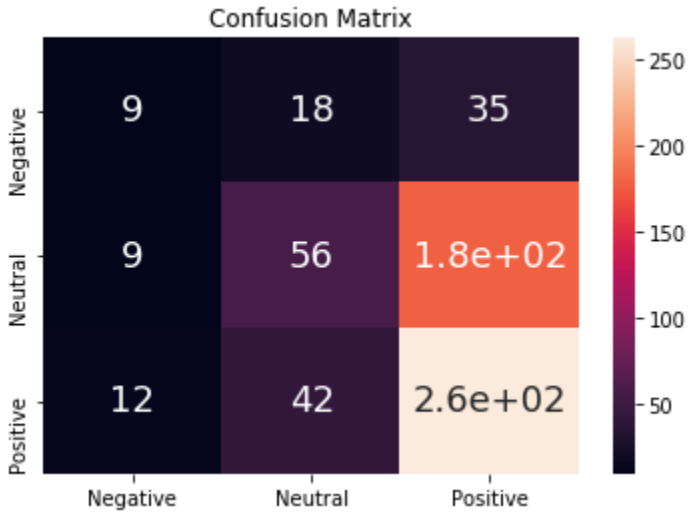
2) NEUTRAL LABEL ANALYSIS :

1 WAS PREDICTED AS NEGATIVE;10 WERE PREDICTED AS NEUTRAL;13 WERE PREDICTED AS POSITIVE.

3) POSITIVE LABEL ANALYSIS :

2 WERE PREDICTED AS NEGATIVE;5 WERE PREDICTED AS NEUTRAL;54 WERE PREDICTED AS POSITIVE.

API SCORES VS ALL THE DATASET :



PREDICTION ANALYSIS :

CORRECT PREDICTION :

NEGATIVE - 9 out of 62 ; NEUTRAL - 56 out of 245 ; POSITIVE - 260 out of 314

1) NEGATIVE LABEL ANALYSIS :

9 WERE PREDICTED AS NEGATIVE; 18 WERE PREDICTED AS NEUTRAL; 35 WERE PREDICTED AS POSITIVE.

2) NEUTRAL LABEL ANALYSIS :

9 WAS PREDICTED AS NEGATIVE;56 WERE PREDICTED AS NEUTRAL;180 WERE PREDICTED AS POSITIVE.

3) POSITIVE LABEL ANALYSIS :

12 WERE PREDICTED AS NEGATIVE;42 WERE PREDICTED AS NEUTRAL;260 WERE PREDICTED AS POSITIVE.

EXPERIMENT 4 :

Auto-ML methods are used in this experiment to train the model , predict the output and calculate the confusion matrix.

TPOT :

The goal of TPOT is to automate the building of ML pipelines by combining a flexible expression tree representation of pipelines.

!pip install tpot – used to install TPOT in Anaconda

After formatting the dataset, train datasets can be fit in the model.

```
tpot = TPOTClassifier(verbosity=3,
                      scoring="accuracy"
                      )

tpot.fit(X_train,y_train)
#print(tpot.score(X_test,y_test))
```

Output :

```
Generation 100 - Current Pareto front scores:
-1      0.6952380952380952      GradientBoostingClassifier(input_matrix, GradientBoostingClassifier__learning_rate=0.01, Grad
ientBoostingClassifier__max_depth=7, GradientBoostingClassifier__max_features=0.6500000000000001, GradientBoostingClassifier_
_min_samples_leaf=1, GradientBoostingClassifier__min_samples_split=17, GradientBoostingClassifier__n_estimators=100, Gradient
BoostingClassifier__subsample=0.8)
-2      0.6976190476190476      RandomForestClassifier(LogisticRegression(input_matrix, LogisticRegression__C=25.0, LogisticR
egression__dual=False, LogisticRegression__penalty=l2), RandomForestClassifier__bootstrap=True, RandomForestClassifier__crite
rion=entropy, RandomForestClassifier__max_features=0.5, RandomForestClassifier__min_samples_leaf=17, RandomForestClassifier__
min_samples_split=17, RandomForestClassifier__n_estimators=100)
-3      0.7238095238095238      GradientBoostingClassifier(PCA(StandardScaler(input_matrix), PCA__iterated_power=9, PCA__svd_
solver=randomized), GradientBoostingClassifier__learning_rate=0.01, GradientBoostingClassifier__max_depth=8, GradientBoosting
Classifier__max_features=0.6500000000000001, GradientBoostingClassifier__min_samples_leaf=1, GradientBoostingClassifier__min_
samples_split=17, GradientBoostingClassifier__n_estimators=100, GradientBoostingClassifier__subsample=0.9000000000000001)
```

The model will take approximately 6 hours to run.

After it is run , we can predict the values to calculate the confusion matrix.

With only Netflix dataset :

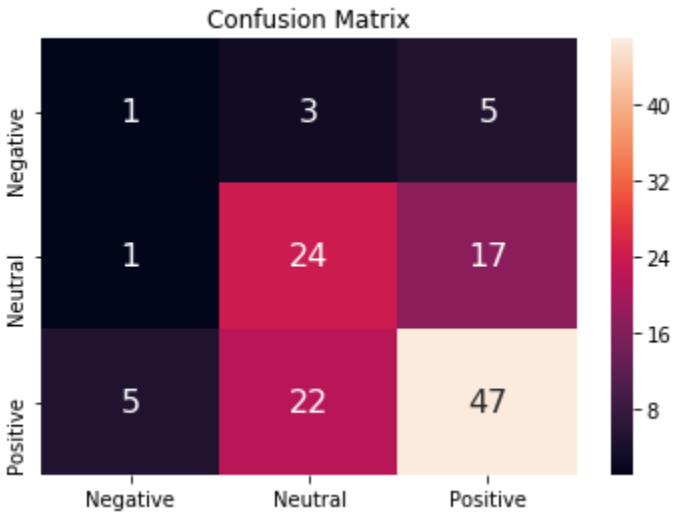
```
array([[ 1,  1,  2],
       [ 0,  3,  6],
       [ 0,  3, 14]], dtype=int64)
```

Accuracy of this model is 0.6

In this positive label is predicted with less error than other two labels.

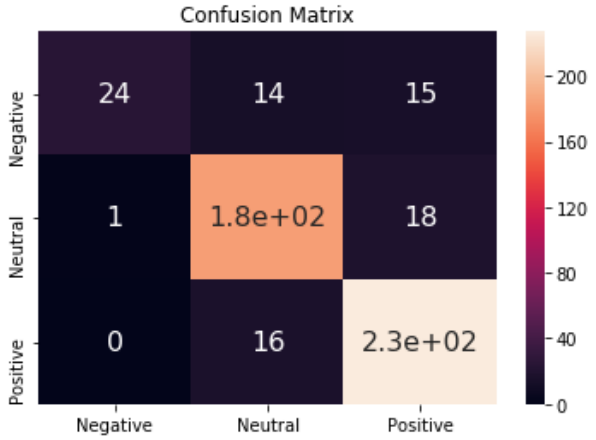
1 out of 4 Negative label, 3 out of 9 Neutral and 14 out of 17 Positive labels are predicted correctly.

With all the dataset :



TEST PREDICTION ANALYSIS :

1 out of 9 Negative labels were predicted correctly ; 24 out of 42 Neutral labels are predicted correctly ; 47 out of 74 Positive labels are predicted correctly.



TRAIN PREDICTION ANALYSIS :

24 out of 53 Negative labels were predicted correctly ; 180 out of 199 Neutral labels are predicted correctly ; 230 out of 246 Positive labels are predicted correctly.

TPOT PREDICTION ANALYSIS :

Negative - 25 out of 62 ; Neutral - 204 out of 241 ; Positive - 277 out of 320 correctly predicted labels

H20.ai :

This is the second Auto-ML method used.

First the dataframe has to be converted to H20dataframe format.

```
d = h2o.H2OFrame(d)
```

```
Parse progress: |████████████████████████████████████████████████████████| 100%
```

After that, just like other models, this dataframe can be split into test and train and fit in model. There are many models provided by H20 similar to SKLEARN library. In this experiment DeepLearningEstimator is used.

NETFLIX DATASET :

```
ModelMetricsMultinomial: deeplearning
** Reported on test data. **

MSE: 0.3859645821333194
RMSE: 0.6212604784897551
LogLoss: 1.5140004282686859
Mean Per-Class Error: 0.5353535353535354
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
```

-1	0	1	Error	Rate
2.0	3.0	1.0	0.6666667	4 / 6
1.0	3.0	5.0	0.6666667	6 / 9
1.0	5.0	16.0	0.2727273	6 / 22
4.0	11.0	22.0	0.4324324	16 / 37

This is the confusion matrix generated for test data. In this the overall error rate is 0.4. The error rate for negative and neutral is high which is 0.6. This model is able to predict positive values correctly with minimal error rate of 0.2.

Here, 2 out of 4 negative labels are labelled correctly, 3 out of 11 labels are labelled correctly and 16 out of 22 labels are labelled correctly.

WITH ALL DATASET :

```
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
```

-1	0	1	Error	Rate
2.0	47.0	3.0	0.9615385	50 / 52
0.0	175.0	22.0	0.1116751	22 / 197
1.0	142.0	98.0	0.5933610	143 / 241
3.0	364.0	123.0	0.4387755	215 / 490

TRAIN PREDICTION ANALYSIS :

Negative - 2 out of 52 correctly predicted labels ; Neutral - 175 out of 197 labels were predicted correctly ; Positive - 98 out of 241 labels were predicted correctly

Confusion Matrix: Row labels: Actual class; Column labels: Predicted class

-1	0	1	Error	Rate
0.0	9.0	1.0	1.0	10 / 10
1.0	38.0	7.0	0.1739130	8 / 46
0.0	46.0	30.0	0.6052632	46 / 76
1.0	93.0	38.0	0.4848485	64 / 132

TEST PREDICTION ANALYSIS

Negative - 0 out of 10 correctly predicted labels ; Neutral - 38 out of 46 labels were predicted correctly ; Positive - 30 out of 76 labels were predicted correctly

H20.ai OVERALL ANALYSIS :

Negative - 2 out of 62 ; Neutral - 213 out of 241 ; Positive - 128 out of 317 were predicted correctly

AUTO-SKLEARN :

Auto Sklearn works only on Linux platform for now.

Classification library from Auto-Sklearn is used for this module.

WITH NETFLIX DATASET :

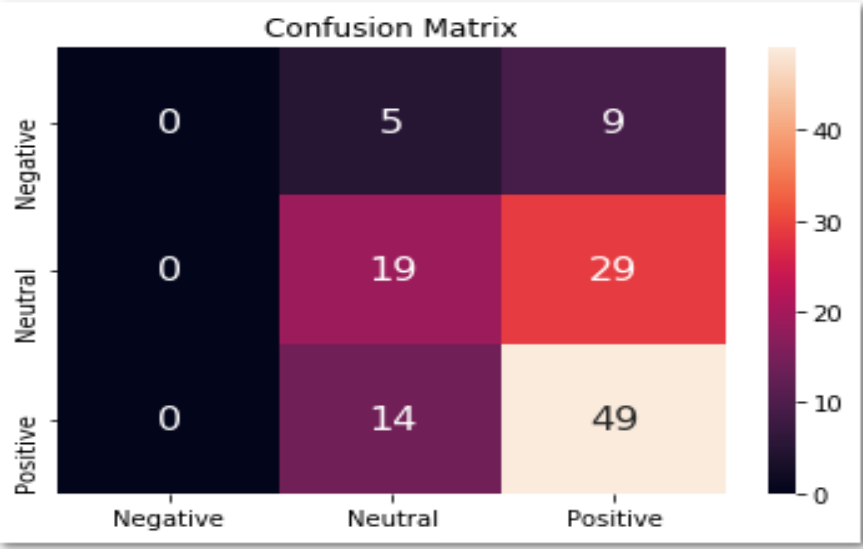
NEGATIVE 0 3 2

NEUTRAL 1 2 1

POSITIVE 1 4 16

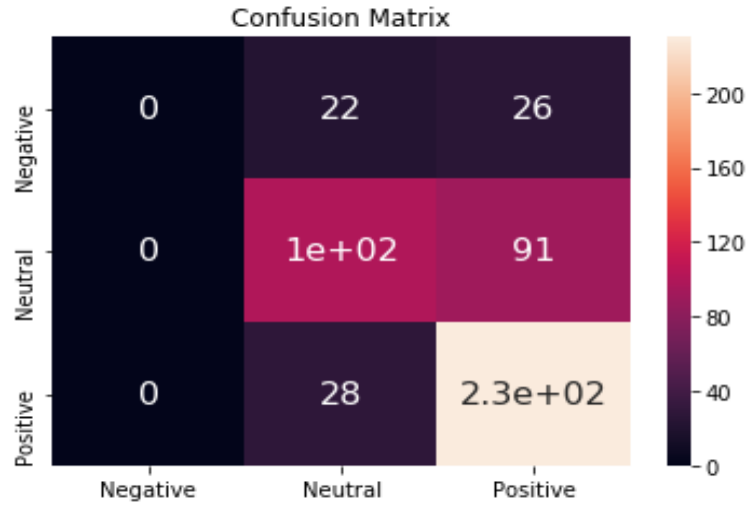
Analysis : 0 out of 5 negative labels were predicted correctly, 2 out of 4 neutral values were predicted correctly, 16 out of 21 positive labels were predicted correctly.

WITH ALL DATASET (TEST):



Analysis : 0 out of 14 negative labels were predicted correctly, 19 out of 48 neutral values were predicted correctly, 49 out of 63 positive labels were predicted correctly.

WITH ALL DATASET(TRAIN):

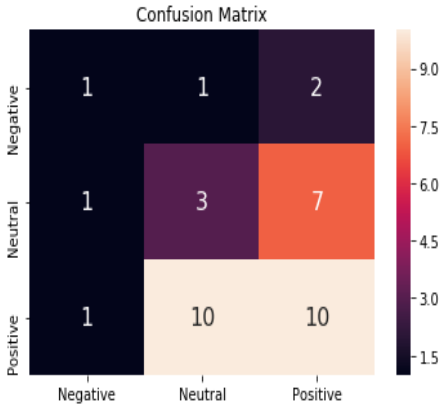


Analysis : 0 out of 48 negative labels were predicted correctly, 100 out of 191 neutral values were predicted correctly, 230 out of 258 positive labels were predicted correctly.

FINAL MODEL :

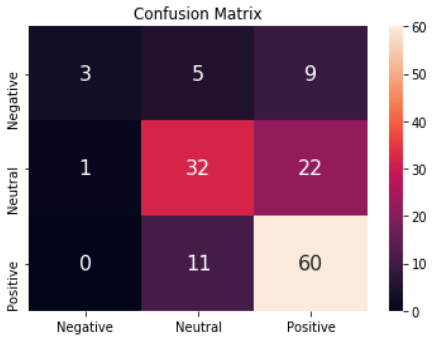
Auto -Sklearn and H20 are rules out because they both do not predict one label which affects the accuracy of the model.

Experiment 3 with API's performed well but TPOT has performed better than that when checking the overall performance.



TEST PREDICTION ANALYSIS :

1 out of 4 Negative labels were predicted correctly ; 3 out of 11 Neutral labels are predicted correctly ;10 out of 21 Positive labels are predicted correctly.



TRAIN PREDICTION ANALYSIS :

3 out of 17 Negative labels were predicted correctly ; 32 out of 55 Neutral labels are predicted correctly ; 60 out of 71 Positive labels are predicted correctly.

TPOT FINAL PREDICTION ANALYSIS :

Negative - 4 out of 21 ; Neutral - 35 out of 66 ; Positive -70 out of 81 correctly predicted labels

REFERENCES :

[a] https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

[b] https://cloud.google.com/natural-language/overview/docs/

[c] https://console.bluemix.net/catalog/services/natural-language-understanding?hideTours=true&?cm_sp=WatsonPlatform-WatsonPlatform-_-OnPageNavCTA-IBMWatson_NaturalLanguageUnderstanding-_-Watson_Developer_Website

[d] https://stackoverflow.com/questions/51697330/confusion-matrix-on-h2o

[e] https://towardsdatascience.com/tpot-automated-machine-learning-in-python-4c063b3e5de9

[f] https://github.com/h2oai/h2o-tutorials/blob/master/h2o-world-2017/automl/README.md

[g] https://automl.github.io/auto-sklearn/stable/