**IST 664**

**Natural Language Processing**

# PROJECT REPORT

***Rashmitha Pandati***

***Rishab Upadhye***

***Aneesh Phatak***

**Table of Contents**

# STOCK MARKET INDICES PREDICTION USING NEWS HEADLINES

## 1. INTRODUCTION:

Stock prices are hard to predict based on some expertise through previous trends or past prices and hence need the help of artificial intelligence and data mining techniques. The volatility of stock prices depends on gains or losses of certain companies. News articles are one of the most important factors which influence the stock market. So for an efficient analysis of the current trends, new company's product information, business growth etc., we propose to look at the daily news which represents factual information about the companies which could be ultimately used to predict the stock prices. Hence, we will be using news articles to predict the change in stock indices rather than predicting the prices by historical stock prices. We plan to perform sentiment analysis of the headlines and understand the investing insight through the emotion behind the headlines and predict whether the market *feels* good or bad about a stock, after which the output will be fed to various machine learning models to predict whether the price of stock indices will go up or not.

## 2. DATA DESCRIPTION:

The dataset consists of 4101 rows and 27 columns where the first column is assigned to Date and second column is assigned to Label. This column consists of binary values where 1 represents if the stock value increased or stayed the same and 0 represents if the stock value decreased. The columns from 3 to 27 consists news headlines ranging from Top 1 to Top 25 corresponding with the respective date mentioned. There are two types of data combined in the dataset:

- **News data:** the data consists of historical news headlines from Reddit World News Channel. They are ranked by reddit users' votes, and only the top 25 headlines are considered for a single date. All the news headlines are ranked from top to bottom based on how *hot* they are.
  (Range: 2008-06-08 to 2016-07-01)
- **Stock data:** Dow Jones Industrial Average (DJIA) is used to "prove the concept". This data has been directly downloaded from the Yahoo Finance website.
  (Range: 2008-08-08 to 2016-07-01)

```
In [397]: df.head()
          #the Label variable will be 1 if the DJIA stayed the same or rose on that date or 0 if the DJIA fell on that date.
```

| Date | Label | Top1 | Top2 | Top3 | Top4 | Top5 | Top6 | Top7 | Top8 | ... | Top16 | Top17 | Top18 | Top19 | Top20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1/3/2000 | 0 | A 'hindrance to operations': extracts from the... | Scorecard | Hughes' instant hit buoys Blues | Jack gets his skates on at ice-cold Alex | Chaos as Maracana builds up for United | Depleted Leicester prevail as Elliott spoils E... | Hungry Spurs sense rich pickings | Gunners so wide of an easy target | ... | Flintoff injury piles on woe for England | Hunters threaten Jospin with new battle of the... | Kohl's successor drawn into scandal | The difference between men and women | Sara Denver, nurse turned solicitor |
| 1/4/2000 | 0 | Scorecard | The best lake scene | Leader: German sleaze inquiry | Cheerio, boyo | The main recommendations | Has Cubie killed fees? | Has Cubie killed fees? | Has Cubie killed fees? | ... | On the critical list | The timing of their lives | Dear doctor | Irish court halts IRA man's extradition to Nor... | Burundi peace initiative fades after rebels re... |
| 1/5/2000 | 0 | Coventry caught on counter by | United's rivals on the road | Thatcher issues defence before | Police help Smith lay down | Tale of Trautmann bears two more | England on the | Pakistan retaliate with call | Cullinan continues his Cape | ... | South Melbourne | Necaxa (Mexico) | Real Madrid | Raja Casablanca | Corinthians (Brazil) |

## 3. MISSING DATA:

The dataset must be checked for missing data and obtain a total count. All the missing data must then be replaced with blank/empty values. After parsing through the data set we find that column 25 has 1 cell with no data, column 26 has 3 cells with no data and column 27 has 3 cells with no data. Hence, we replace the following missing values with a blank/empty value.

```
In [400]: # Counting missing values
          df.isnull().sum()

Out[400]: Date      0
          Label     0
          Top1      0
          Top2      0
          Top3      0
          Top4      0
          Top5      0
          Top6      0
          Top7      0
          Top8      0
          Top9      0
          Top10     0
          Top11     0
          Top12     0
          Top13     0
          Top14     0
          Top15     0
          Top16     0
          Top17     0
          Top18     0
          Top19     0
          Top20     0
          Top21     0
          Top22     0
          Top23     1
          Top24     3
          Top25     3
          dtype: int64
```

```
In [401]: #Replacing missing values with a blank
          df = df.replace(np.nan, ' ', regex=True)

          #double check
          df.isnull().sum().sum()

Out[401]: 0
```

## 4. COMBINING THE NEWS HEADLINES:

To make the classification and prediction process easier we combine all the 25 news headlines in a new column called "Combine" with respective to their dates. We also assign the values in the Label column of 1 to UP variable and 0 to DOWN variable.

```
In [402]: #Adding a column 'Combined' which contains all the top 25 headlines in one cell for each row.
          df['Combined']=df.iloc[:,2:27].apply(lambda row: ''.join(str(row.values)),axis=1)

In [403]: up = df[df['Label']==1]
          down = df[df['Label']==0]
          print(len(up)/len(df))

          0.5281638624725676
```

## 5. DATA PREPROCESSING:

It is an essential step in Natural Language Processing as depending on how well the data has been preprocessed the results are seen. In this project the following pre-processing steps have been considered based on the context and the necessity of the data:

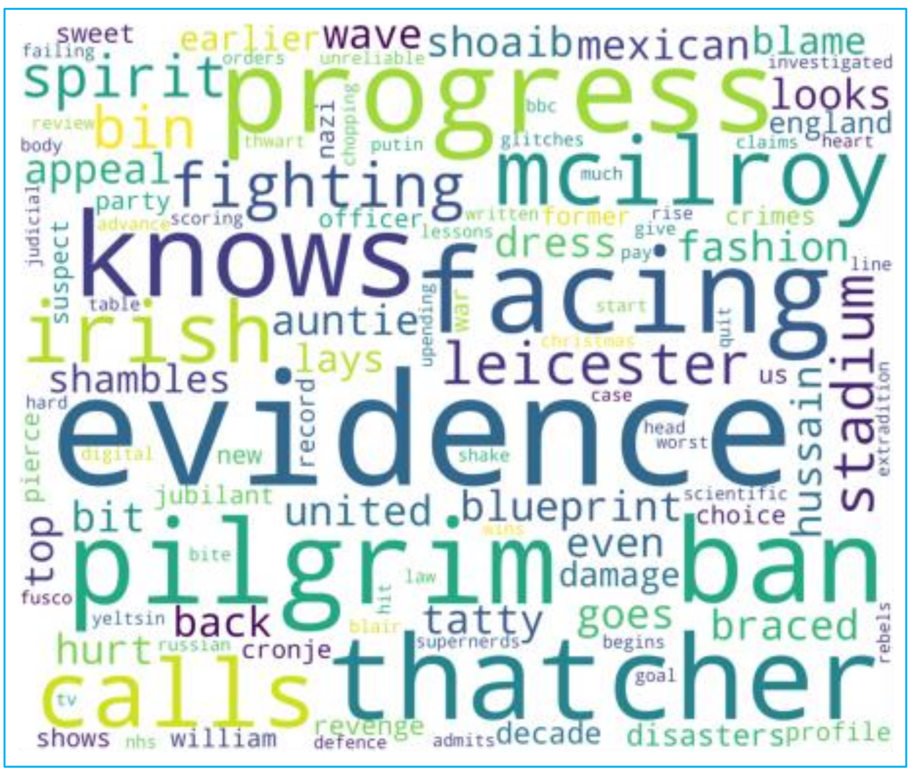
- Removing Alpha Numeric Characters
- Lowercasing all the words
- Removing all the stop words

```
In [404]: def to_words(content):
              letters = re.sub("[^a-zA-Z]"," ", content)
              words = letters.lower().split()
              stops = set(stopwords.words("english"))
              mwords = [w for w in words if not w in stops]
              return(" ".join( mwords))
          up_word = []
          down_word = []
          for word in up['Combined']:
              up_word.append(to_words(word))

          for word in down['Combined']:
              down_word.append(to_words(word))

In [405]: down_word[0:5]

Out[405]: ['hindrance operations extracts leaked reports scorecard hughes instant hit buoys blues jack gets skates ice cold alex chaos ma
          racana builds united depleted leicester prevail elliott spoils everton party hungry spurs sense rich pickings gunners wide easy
          target derby raise glass strupar debut double southgate strikes leeds pay penalty hammers hand robson youthful lesson saints pa
          rty like wear wolves turned lambs stump mike catches testy gough taunt langer escapes hit flintoff injury piles woe england hun
          ters threaten jospin new battle somme kohl successor drawn scandal difference men women sara denver nurse turned solicitor dian
          a landmine crusade put tories panic yeltsin resignation caught opposition flat footed russian roulette sold recovering title',
           'scorecard best lake scene leader german sleaze inquiry cheerio boyo main recommendations cubie killed fees cubie killed fees
          cubie killed fees hopkins furious foster lack hannibal appetite cubie killed fees tale two tails say like like say elbows eyes
          nipples task force assess risk asteroid collision found last critical list timing lives dear doctor irish court halts ira man e
          xtradition northern ireland burundi peace initiative fades rebels reject mandela mediator pe points way forward ecb campaigners
          keep pressure nazi war crimes suspect jane ratcliffe yet things know without movies millennium bug fails bite',
           'coventry caught counter flo united rivals road rio thatcher issues defence trial video police help smith lay law everton tale
          trautmann bears two retellings england rack pakistan retaliate call video walsh cullinan continues cape monopoly mcgrath puts i
          ndia misery blair witch bandwagon rolls pele turns heat ferguson party divided kohl slush fund scandal manchester united englan
          d women record south pole walk vasco da gama brazil south melbourne australia necaxa mexico real madrid spain raja casablanca m
```

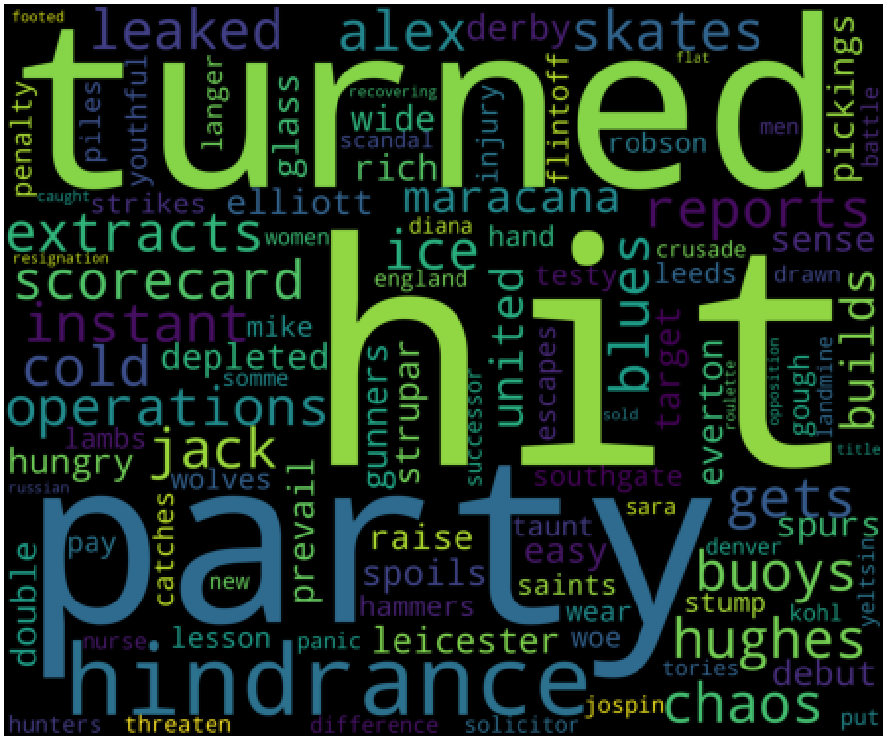
## 6. GENERATION OF WORD CLOUD:

Word clouds are used to easily produce a summary of large documents to visualize data. The novelty visual representation of text data allows us to get high-level information about the current analyzed document. Tags are usually single words, and the importance of each tag is shown with font size or color. This format is useful for quickly perceiving the most prominent terms to determine its relative prominence. Following is the word cloud generated by the analysis:

a. **For Top UP [Positive] News Headlines:**



## b. For Top DOWN [Negative] News Headlines:




## 7. DATA SPLITTING:

Separating data into training and testing sets is an important part of evaluating data mining models. Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing. Analysis Services randomly samples the data to help ensure that the testing and training sets are similar. By using similar data for training and testing, you can minimize the effects of data discrepancies and better understand the characteristics of the model. After a model has been processed by using the training set, you test the model by making predictions against the test set. Because the data in the testing set already contains known values for the attribute that you want to predict, it is easy to determine whether the model's guesses are correct. Here we used 80% of the data for training and 20% for testing.

```
In [411]: #Splitting data into train and test
          train, test = train_test_split(df,train_size=0.8,test_size=0.2,random_state=1)

In [412]: train.shape

Out[412]: (3280, 28)

In [413]: test.shape

Out[413]: (821, 28)

In [414]: #All headlines in a row combined for train dataset
          trainheadlines = []
          for row in range(0,len(train.index)):
              trainheadlines.append(' '.join(str(x) for x in train.iloc[row,2:27]))

In [415]: #All headlines in a row combined for test dataset
          testheadlines = []
          for row in range(0,len(test.index)):
              testheadlines.append(' '.join(str(x) for x in test.iloc[row,2:27]))
```

## 8. ENCODING THE DATA:

Text data requires special preparation before we can start using it for predictive modeling. The data must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating-point values to use then as an input to machine learning algorithms called Vectorization. The scikit learn library offers easy-to-use tools to perform both tokenization and vectorization of the text data. The Count Vectorizer provides a simple way to both tokenize a collection of text document and build a vocabulary of known words but also to encode new documents using that vocabulary. An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

```
In [417]: #Scikit-Learn’s CountVectorizer is used to convert a collection of text documents to a vector of term/token counts.
          #n=1 for this n-gram model
          basicvectorizer = CountVectorizer()
          basictrain = basicvectorizer.fit_transform(trainheadlines)
          print(basictrain.shape)

          (3280, 44084)

In [418]: print(basicvectorizer.get_feature_names())

          ['00', '000', '00021', '000bpd', '000ft', '000kg', '000km2', '000m', '000mph', '000rmb', '000s', '000th', '000x', '001', '00
          4', '007', '00am', '00pm', '01', '017', '018', '02', '020', '0220', '0221', '03', '035', '037', '04', '045', '04am', '05', '0
          6', '07', '077', '08', '080', '089', '09', '0900', '0930', '094', '10', '100', '1000', '10000', '1000km', '1000s', '1000th',
          '1006', '100bn', '100ds', '100ft', '100k', '100km', '100m', '100mb', '100mil', '100mw', '100s', '100th', '100x', '101', '101s
```

## 9. RANDOM FOREST CLASSIFIER:

Random Forest Classifier creates a set of decision trees from randomly selected subset of the training set. It then aggregates the different votes from different decision trees to decide the final class of the test object. It is an ensemble tree-based learning algorithm. Ensemble algorithms are those which **combines more than one algorithm of same or different kind for classifying objects**. The major advantage of this classifier is that it gives estimates of what variables that are important in the classification. It generates an internal **unbiased estimate of the generalization error** as the forest building progresses and can also **handle thousands of input variables** without variable deletion.

## 10. LOGISTIC REGRESSION CLASSIFIER:

The coefficients of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation. Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data. The best coefficients would result in a model that would predict a value very close to 1 for the default class and a value very close to 0 for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients that minimize the error in the probabilities predicted by the model to those in the data.

## 11. N-GRAM MODEL:

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words, or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called as Shingles. An n-gram model is a probabilistic language model used for predicting the next time in a sequence of the form (n-1) order Markov model. In this project we have run the Unigram, Bigram and Trigram Models using Random Forest and Logistic Regression classifiers to compare the accuracy of our predictions.

a. **UNI-GRAM MODEL:**

According to Latin numerical prefixes, an n-gram of size 1 is referred to as “Unigram”.

```
In [418]: print(basicvectorizer.get_feature_names())

['00', '000', '00021', '000bpd', '000ft', '000kg', '000km2', '000m', '000mph', '000rmb', '000s', '000th', '000x', '001', '00
4', '007', '00am', '00pm', '01', '017', '018', '02', '020', '0220', '0221', '03', '035', '037', '04', '045', '04am', '05', '0
6', '07', '077', '08', '080', '089', '09', '0900', '0930', '094', '10', '100', '1000', '10000', '1000km', '1000s', '1000th',
'1006', '100bn', '100ds', '100ft', '100k', '100km', '100m', '100mb', '100mil', '100mw', '100s', '100th', '100x', '101', '101s
t', '102', '103', '1030', '1038', '104', '1044', '104m', '105', '1050', '1054', '106', '1061c', '106fm', '106m', '107', '107
d', '108', '109', '10bars', '10bn', '10cm', '10in', '10k', '10km', '10m', '10million', '10s', '10th', '10x', '10yr', '11', '1
10', '110m', '111', '111m', '112', '113', '114', '1142', '115', '115m', '116', '116bn', '117', '117k', '117m', '118', '119',
'119th', '11am', '11b', '11bn', '11c', '11g', '11m', '11s', '11th', '11yo', '12', '120', '1200', '120bn', '120ft', '120m', '1
21', '1215', '122', '122f', '123', '124', '125', '1255', '125bn', '125m', '125mph', '126', '127', '1270', '127bn', '128', '12
8kbps', '129', '12b', '12bn', '12k', '12m', '12million', '12th', '12tn', '13', '130', '1300', '1300gmt', '130bn', '130mph',
'131', '1310', '133', '134', '135', '1351', '136', '137', '138', '138bn', '139', '13bn', '13km', '13m', '13th', '14', '140',
'1400', '1400s', '1408', '140bn', '141', '142', '1425', '143', '144', '1440', '145', '1466a', '147', '148', '1481', '1485',
'148m', '149', '14bn', '14c', '14m', '14th', '14x', '14yo', '15', '150', '1500', '150ft', '150m', '150th', '151', '152', '15
3', '1536', '153bn', '154', '15431292', '155', '156', '1562', '1566', '157', '158', '15bn', '15gbp', '15m', '15mar2011', '15p
m', '15s', '15th', '15yo', '16', '160', '1600', '16000', '160m', '160mph', '161', '1610', '1615', '162', '1623', '163', '163
m', '164', '1642', '165', '166', '1667', '166th', '167', '1677', '1679', '168', '16bn', '16m', '16s', '16th', '16thcentury',
'17', '170', '1700', '17000', '17000km', '1707', '172', '172b', '172m', '172mph', '173', '175', '1752', '1759', '176', '177',
'1773', '1775', '1776', '178', '1780', '1782', '179', '17bn', '17m', '17p', '17th', '18', '180', '1800kg', '180mb', '181', '1
812', '182', '1828', '183', '1830', '1839', '184', '1840s', '1842', '1843', '185', '1850', '186', '1860', '1873', '187m', '18
```

i. **UNIGRAM MODEL TOP 5 POSITIVE WORDS:**

The top 5 positive words along with their coefficients for the Unigram Model are as follows:

| | Word | Coefficient |
|---|---|---|
| **1552** | abroad | 0.714692 |
| **33098** | resolution | 0.697633 |
| **41906** | verdict | 0.675517 |
| **19072** | hospital | 0.615658 |
| **27440** | northern | 0.615633 |

ii. **UNIGRAM MODEL TOP 5 NEGATIVE WORDS:**

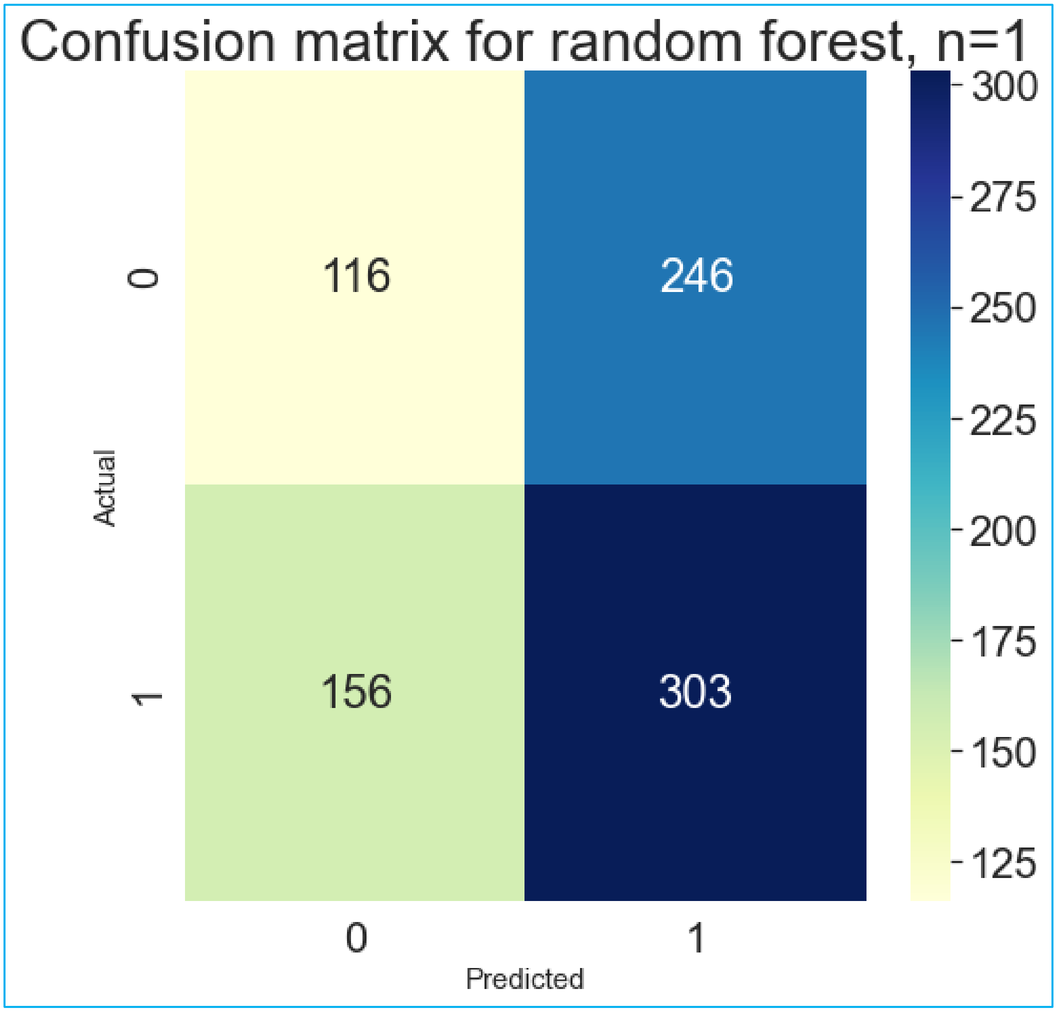
The top 5 negative words along with their coefficients for the Unigram Model are as follows:

| | Word | Coefficient |
|---|---|---|
| **200** | 15 | -0.631947 |
| **3918** | avoid | -0.657068 |
| **38733** | system | -0.695755 |
| **14345** | extra | -0.699330 |
| **39191** | tell | -0.701835 |

### iii. UNI-GRAM MODEL USING RANDOM FOREST:

```
#n=1, random forest
RFmodel1=RandomForestClassifier(n_estimators=200,criterion='entropy')
RFmodel1=RFmodel1.fit(basictrain,train['Label'])

In [422]: basictest = basicvectorizer.transform(testheadlines)
          preds1 = RFmodel1.predict(basictest)
          acc1=accuracy_score(test['Label'], preds1)

In [423]: print('Random Forest Accuracy: ',acc1 )

          Logistic Regression 1 accuracy:  0.5054811205846529

In [424]: matrix=confusion_matrix(test['Label'],preds1)
          print(matrix)
          score=accuracy_score(test['Label'],preds1)
          print(score)
          report=classification_report(test['Label'],preds1)
          print(report)

          [[104 258]
           [148 311]]
          0.5054811205846529
                        precision    recall  f1-score   support

                     0       0.41      0.29      0.34       362
                     1       0.55      0.68      0.61       459

              accuracy                           0.51       821
             macro avg       0.48      0.48      0.47       821
          weighted avg       0.49      0.51      0.49       821
```

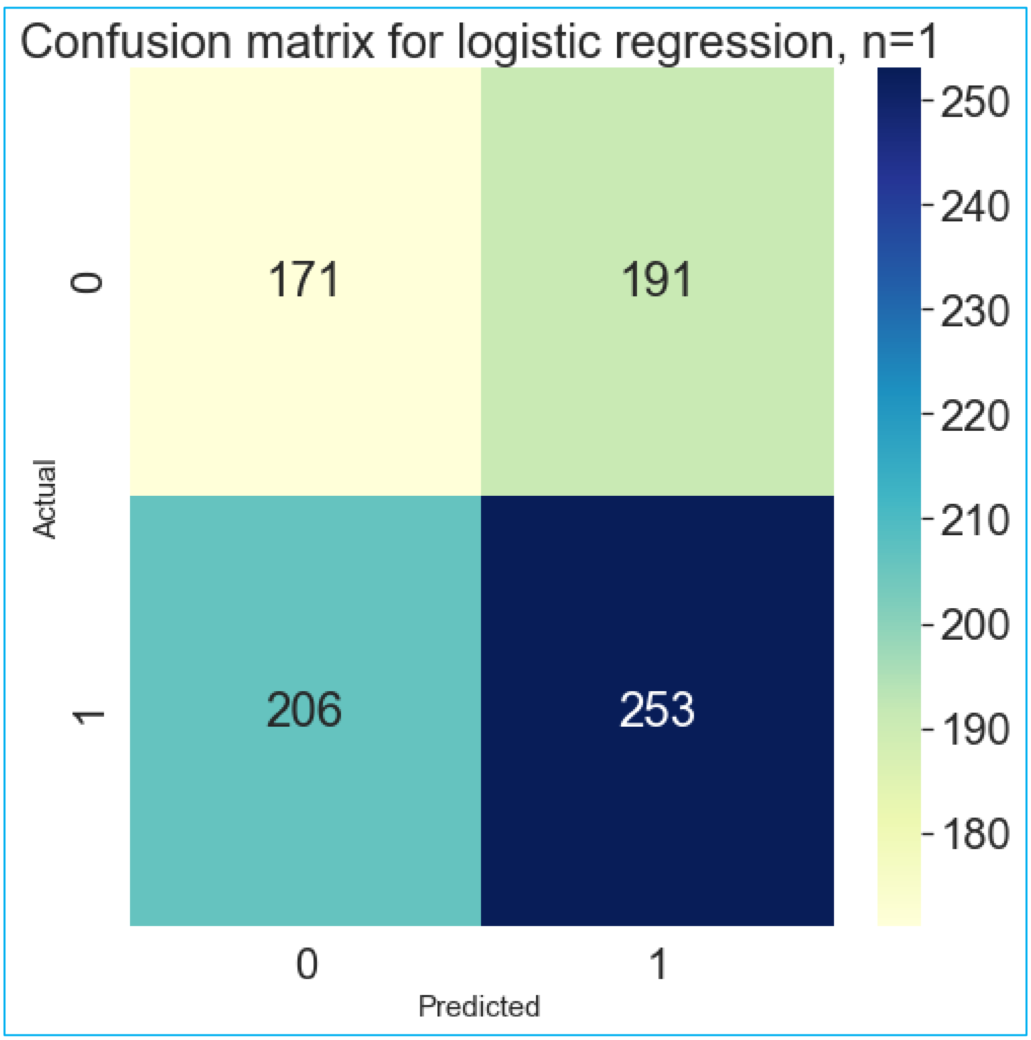
❖ **UNI-GRAM RANDOM FOREST CONFUSION MATRIX:**



## iv. UNI-GRAM MODEL USING LOGISTIC REGRESSION:

```
In [425]: #n=1,logistic regression
          LRmodel1 = LogisticRegression(max_iter=1000)
          LRmodel1 = LRmodel1.fit(basictrain, train["Label"])

In [426]: basictest = basicvectorizer.transform(testheadlines)
          preds2 = LRmodel1.predict(basictest)
          acc2=accuracy_score(test['Label'], preds2)

In [427]: matrix=confusion_matrix(test['Label'],preds2)
          print(matrix)
          score=accuracy_score(test['Label'],preds2)
          print(score)
          report=classification_report(test['Label'],preds2)
          print(report)

          [[171 191]
           [206 253]]
          0.5164433617539586
                        precision    recall  f1-score   support

                     0       0.45      0.47      0.46       362
                     1       0.57      0.55      0.56       459

              accuracy                           0.52       821
             macro avg       0.51      0.51      0.51       821
          weighted avg       0.52      0.52      0.52       821
```

### ❖ UNI-GRAM LOGISTIC REGRESSION CONFUSION MATRIX:



b. **BI-GRAM MODEL:**

According to Latin numerical prefixes, an n-gram of size 2 is referred to as “Bigram”.

```
In [432]: print(advancedvectorizer.get_feature_names())

['000 people', 'about the', 'access to', 'according to', 'accused of', 'across the', 'after being', 'after the', 'against the',
'ahead of', 'al jazeera', 'al qaeda', 'all the', 'and other', 'and the', 'are being', 'are not', 'around the', 'arrested in',
'as the', 'at least', 'at the', 'attack on', 'attempt to', 'back to', 'ban on', 'bbc news', 'be the', 'because of', 'by the',
'call for', 'calls for', 'can be', 'charged with', 'city of', 'climate change', 'could be', 'country diary', 'david cameron',
'dead in', 'due to', 'during the', 'edward snowden', 'end of', 'european union', 'first time', 'for his', 'for the', 'forced t
o', 'found in', 'from the', 'go to', 'going to', 'guilty of', 'has been', 'have been', 'have to', 'he was', 'head of', 'hit b
y', 'hong kong', 'horse racing', 'how the', 'how to', 'human rights', 'hundreds of', 'if they', 'if you', 'in afghanistan', 'in
an', 'in brief', 'in china', 'in egypt', 'in europe', 'in gaza', 'in germany', 'in his', 'in india', 'in iraq', 'in israel', 'i
n its', 'in mexico', 'in new', 'in pakistan', 'in prison', 'in russia', 'in south', 'in syria', 'in the', 'in their', 'into th
e', 'is being', 'is no', 'is not', 'is now', 'is the', 'is to', 'islamic state', 'it has', 'it is', 'it was', 'it will', 'jaile
d for', 'killed by', 'killed in', 'kim jong', 'last year', 'leader in', 'likely to', 'linked to', 'manchester united', 'market
forces', 'may be', 'may have', 'members of', 'middle east', 'millions of', 'more than', 'new york', 'new zealand', 'news in',
'no longer', 'north korea', 'north korean', 'not be', 'not to', 'number of', 'of all', 'of an', 'of his', 'of its', 'of new',
'of people', 'of the', 'of their', 'of thousands', 'of us', 'off the', 'on monday', 'on the', 'on thursday', 'on tuesday', 'on
wednesday', 'one of', 'out of', 'over the', 'part of', 'people in', 'percent of', 'pick of', 'plan to', 'plans to', 'prime mini
ster', 'ready to', 'refuses to', 'return to', 'review the', 'right to', 'role in', 'round up', 'rugby league', 'rugby union',
'said on', 'saudi arabia', 'says he', 'says it', 'sentenced to', 'set to', 'shot dead', 'should be', 'since the', 'south afric
a', 'south korea', 'state of', 'that it', 'that the', 'the best', 'the british', 'the city', 'the country', 'the day', 'the en
d', 'the european', 'the first', 'the government', 'the internet', 'the last', 'the middle', 'the most', 'the new', 'the next',
'the past', 'the right', 'the same', 'the streets', 'the uk', 'the un', 'the united', 'the us', 'the war', 'the way', 'the wes
t', 'the world', 'their own', 'there is', 'they are', 'they were', 'this is', 'this year', 'thousands of', 'threat to', 'threat
ens to', 'to ban', 'to be', 'to build', 'to cut', 'to death', 'to do', 'to end', 'to fight', 'to get', 'to give', 'to go', 'to
have', 'to help', 'to join', 'to keep', 'to kill', 'to leave', 'to make', 'to pay', 'to prevent', 'to protect', 'to save', 'to
stay', 'to stop', 'to take', 'to the', 'to use', 'trying to', 'united nations', 'united states', 'up for', 'up in', 'up the',
'up to', 'us and', 'use of', 'vladimir putin', 'want to', 'wants to', 'war crimes', 'war on', 'way to', 'we are', 'west bank',
'what the', 'who was', 'will be', 'will not', 'with the', 'world cup', 'would be', 'year old', 'years ago', 'years in']
```

i. **BIGRAM MODEL TOP 5 POSITIVE WORDS:**

The top 5 positive words along with their coefficients for the Bigram Model are as follows:

| | Word | Coefficient |
|---|---|---|
| 13 | and other | 1.239008 |
| 49 | found in | 1.185589 |
| 263 | year old | 1.140398 |
| 214 | to build | 1.083737 |
| 98 | it has | 1.048133 |

ii. **BIGRAM MODEL TOP 5 NEGATIVE WORDS:**

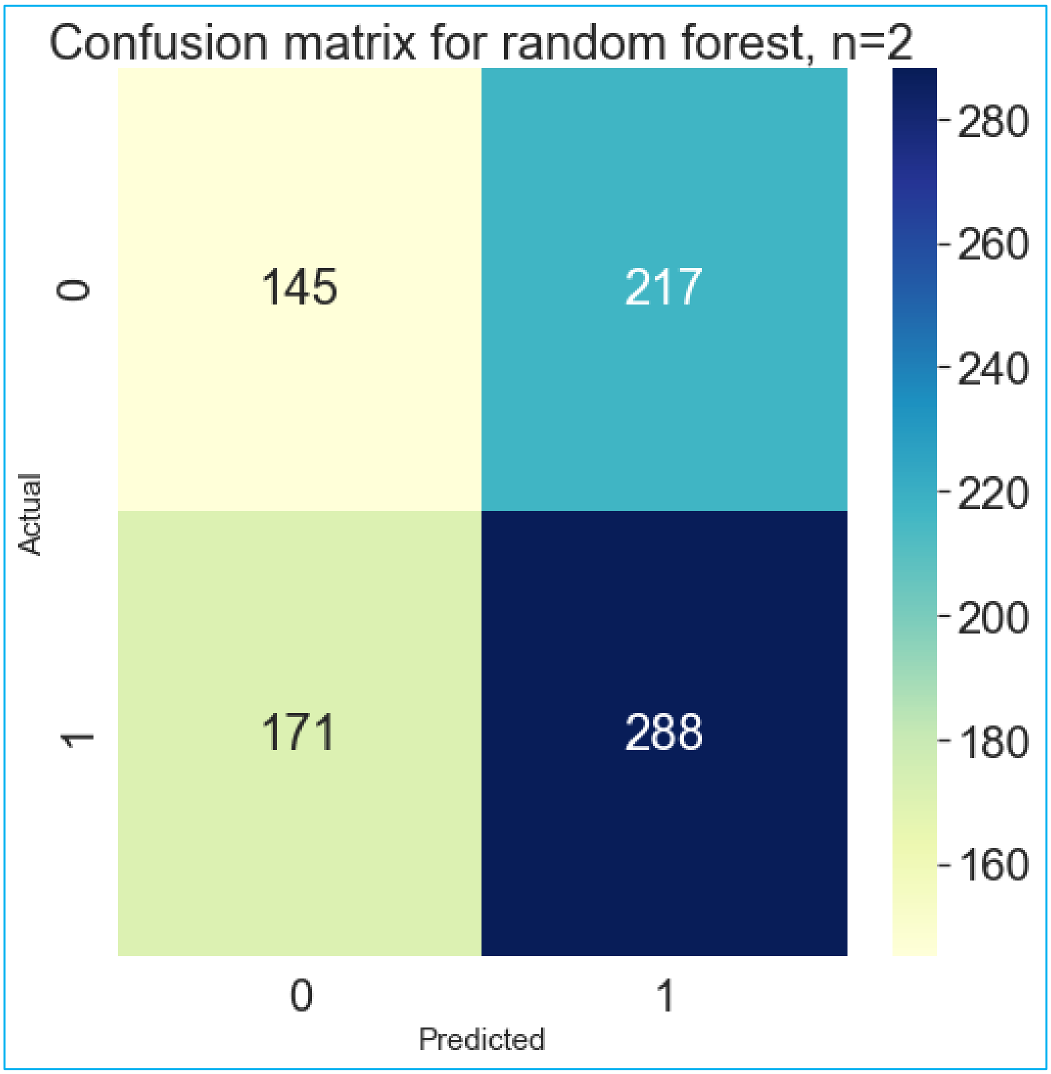
The top 5 negative words along with their coefficients for the Bigram Model are as follows:

| | Word | Coefficient |
|---|---|---|
| 101 | it will | -1.084673 |
| 264 | years ago | -1.111348 |
| 85 | in russia | -1.168715 |
| 244 | up the | -1.227317 |
| 138 | on monday | -1.272808 |

### iii. BI-GRAM MODEL USING RANDOM FOREST:

```
In [433]: #Random forest, n=2
          RFmodel2=RandomForestClassifier(n_estimators=500,criterion='entropy')
          RFmodel2=RFmodel2.fit(advancedtrain,train['Label'])

In [434]: advancedtest = advancedvectorizer.transform(testheadlines)
          preds3 = RFmodel2.predict(advancedtest)
          acc3=accuracy_score(test['Label'], preds3)

In [435]: matrix=confusion_matrix(test['Label'],preds3)
          print(matrix)
          score=accuracy_score(test['Label'],preds3)
          print(score)
          report=classification_report(test['Label'],preds3)
          print(report)

[[137 225]
 [179 280]]
0.5079171741778319
              precision    recall  f1-score   support

           0       0.43      0.38      0.40       362
           1       0.55      0.61      0.58       459

    accuracy                           0.51       821
   macro avg       0.49      0.49      0.49       821
weighted avg       0.50      0.51      0.50       821
```

❖ **BI-GRAM RANDOM FOREST CONFUSION MATRIX:**



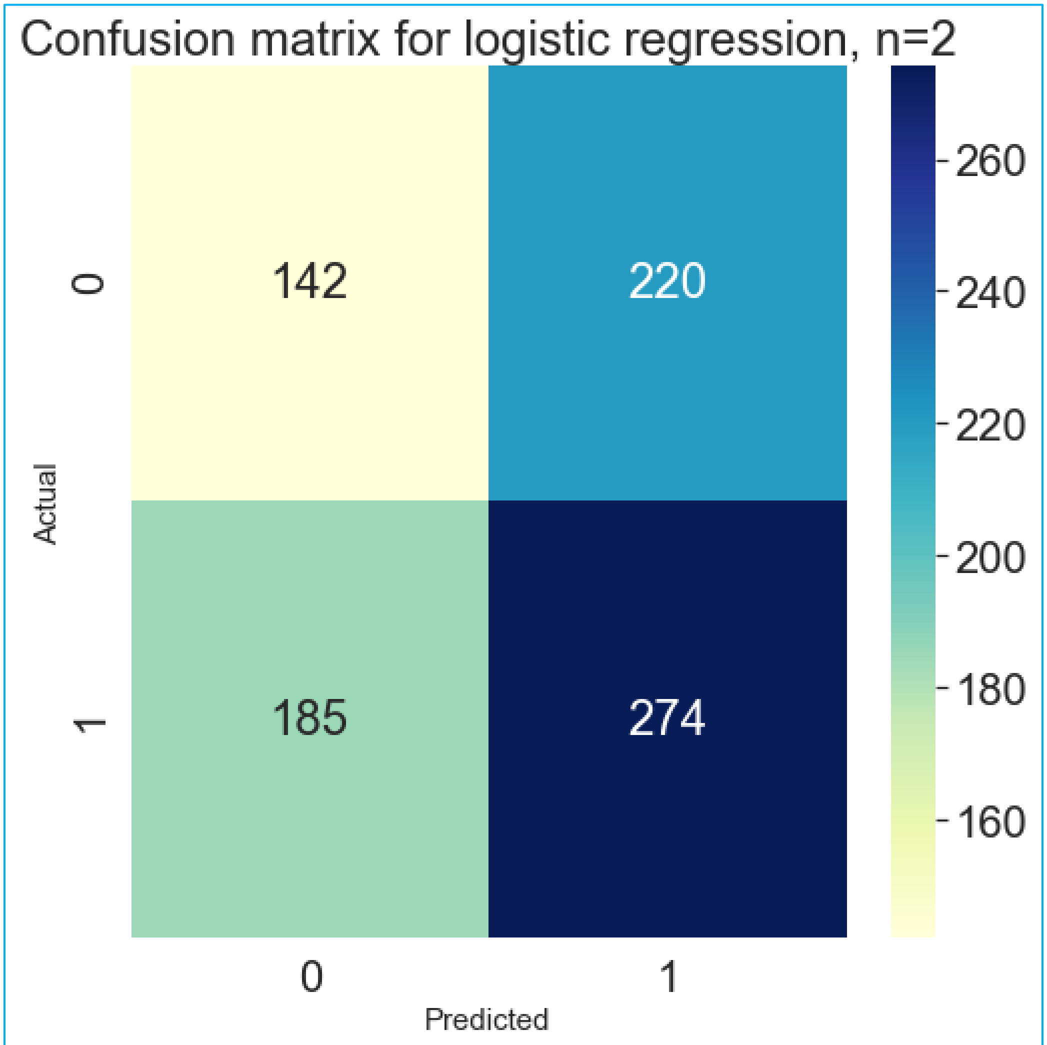
## iv. BI-GRAM MODEL USING LOGISTIC REGRESSION:

```
In [437]: #n=2, Logistic regression
LRmodel2 = LogisticRegression(max_iter=1000)
LRmodel2 = LRmodel2.fit(advancedtrain, train["Label"])


advancedtest = advancedvectorizer.transform(testheadlines)
preds4 = LRmodel2.predict(advancedtest)
acc4=accuracy_score(test['Label'], preds4)
```

```
In [438]: matrix=confusion_matrix(test['Label'],preds4)
print(matrix)
score=accuracy_score(test['Label'],preds4)
print(score)
report=classification_report(test['Label'],preds4)
print(report)
```

```
[[142 220]
 [185 274]]
0.5066991473812423
              precision    recall  f1-score   support

           0       0.43      0.39      0.41       362
           1       0.55      0.60      0.58       459

    accuracy                           0.51       821
   macro avg       0.49      0.49      0.49       821
weighted avg       0.50      0.51      0.50       821
```

- ❖ **BI-GRAM LOGISTIC REGRESSION CONFUSION MATRIX:**



c. **TRI-GRAM MODEL:**

According to Latin numerical prefixes, an n-gram of size 3 is referred to as “Trigram”.

```
In [443]: print(advancedvectorizer2.get_feature_names())

['000 people have', '000 year old', '11 year old', '12 year old', '13 year old', '14 year old', '15 year old', '16 year old',
'17 year old', '2022 world cup', 'access to the', 'according to new', 'according to report', 'according to the', 'across the co
untry', 'ahead of the', 'al jazeera english', 'albert hall london', 'an act of', 'an attack on', 'an attempt to', 'an effort t
o', 'an end to', 'and forced to', 'and human rights', 'and south korea', 'and that the', 'and the united', 'and the us', 'and t
he world', 'anniversary of the', 'are expected to', 'are trying to', 'around the world', 'as long as', 'as many as', 'as much a
s', 'as one of', 'as part of', 'as result of', 'as well as', 'as young as', 'at least 10', 'at risk of', 'at the end', 'at the
top', 'at the united', 'attack on the', 'aung san suu', 'australian prime minister', 'back in the', 'back to the', 'ban ki moo
n', 'bank of england', 'bank of scotland', 'bashar al assad', 'be able to', 'be allowed to', 'be banned from', 'be forced to',
'be the first', 'be used to', 'beaten to death', 'because of the', 'become the first', 'been accused of', 'been arrested in',
'been found in', 'been killed in', 'been sentenced to', 'beginning of the', 'believed to be', 'believed to have', 'billions of
dollars', 'british prime minister', 'business news in', 'by end of', 'by the end', 'by the government', 'by the police', 'by th
e united', 'by the us', 'can no longer', 'central african republic', 'chancellor angela merkel', 'child sex abuse', 'claims to
have', 'control of the', 'corrections and clarifications', 'could be the', 'could lead to', 'countries in the', 'country diary
wenlock', 'country in the', 'court has ruled', 'crack down on', 'cracks down on', 'crimes against humanity', 'dark side of', 'd
eath penalty for', 'deputy prime minister', 'diary wenlock edge', 'division round up', 'do not have', 'doctors without border
s', 'drone strike kills', 'due to the', 'end of the', 'end to the', 'england and wales', 'european court of', 'european round u
p', 'expected to be', 'female genital mutilation', 'festival hall london', 'first time in', 'first time since', 'foot and mout
h', 'for end to', 'for failing to', 'for first time', 'for human rights', 'for more than', 'for refusing to', 'for the first',
'for the past', 'for the us', 'for trying to', 'for two years', 'for up to', 'for war crimes', 'found dead in', 'found guilty o
f', 'found in the', 'founder julian assange', 'freedom of expression', 'freedom of speech', 'from north korea', 'from the inter
net', 'fukushima nuclear plant', 'full text of', 'german chancellor angela', 'go to the', 'going to be', 'great barrier reef',
'gulf of mexico', 'half of the', 'has agreed to', 'has become the', 'has been arrested', 'has been found', 'has been sentence
d', 'has called for', 'has decided to', 'has ruled that', 'has said that', 'has threatened to', 'have been arrested', 'have bee
n found', 'have been killed', 'have died in', 'have the right', 'have to be', 'head of the', 'how to be', 'howlett film picks',
'human rights abuses', 'human rights council', 'human rights groups', 'human rights lawyer', 'human rights watch', 'hume market
forces', 'hundreds of thousands', 'if they are', 'in afghanistan the', 'in an attack', 'in an attempt', 'in an effort', 'in bid
to', 'in charge of', 'in danger of', 'in east jerusalem', 'in eastern ukraine', 'in england and', 'in europe and', 'in exchange
for', 'in favor of', 'in favour of', 'in front of', 'in hong kong', 'in iraq and', 'in jail for', 'in less than', 'in line fo
r', 'in middle east', 'in more than', 'in new york', 'in new zealand', 'in north korea', 'in northern ireland', 'in one of', 'i
n order to', 'in praise of', 'in prison for', 'in recent years', 'in response to', 'in saudi arabia', 'in search of', 'in south
```

i. **TRIGRAM MODEL TOP 5 POSITIVE WORDS:**

The top 5 positive words along with their coefficients for the Trigram Model are as follows:

| | Word | Coefficient |
|---|---|---|
| 3 | 12 year old | 1.239925 |
| 248 | in west bank | 1.221825 |
| 116 | first time since | 1.212255 |
| 631 | this is not | 1.209642 |
| 588 | the start of | 1.129639 |

ii. **TRIGRAM MODEL TOP 5 NEGATIVE WORDS:**

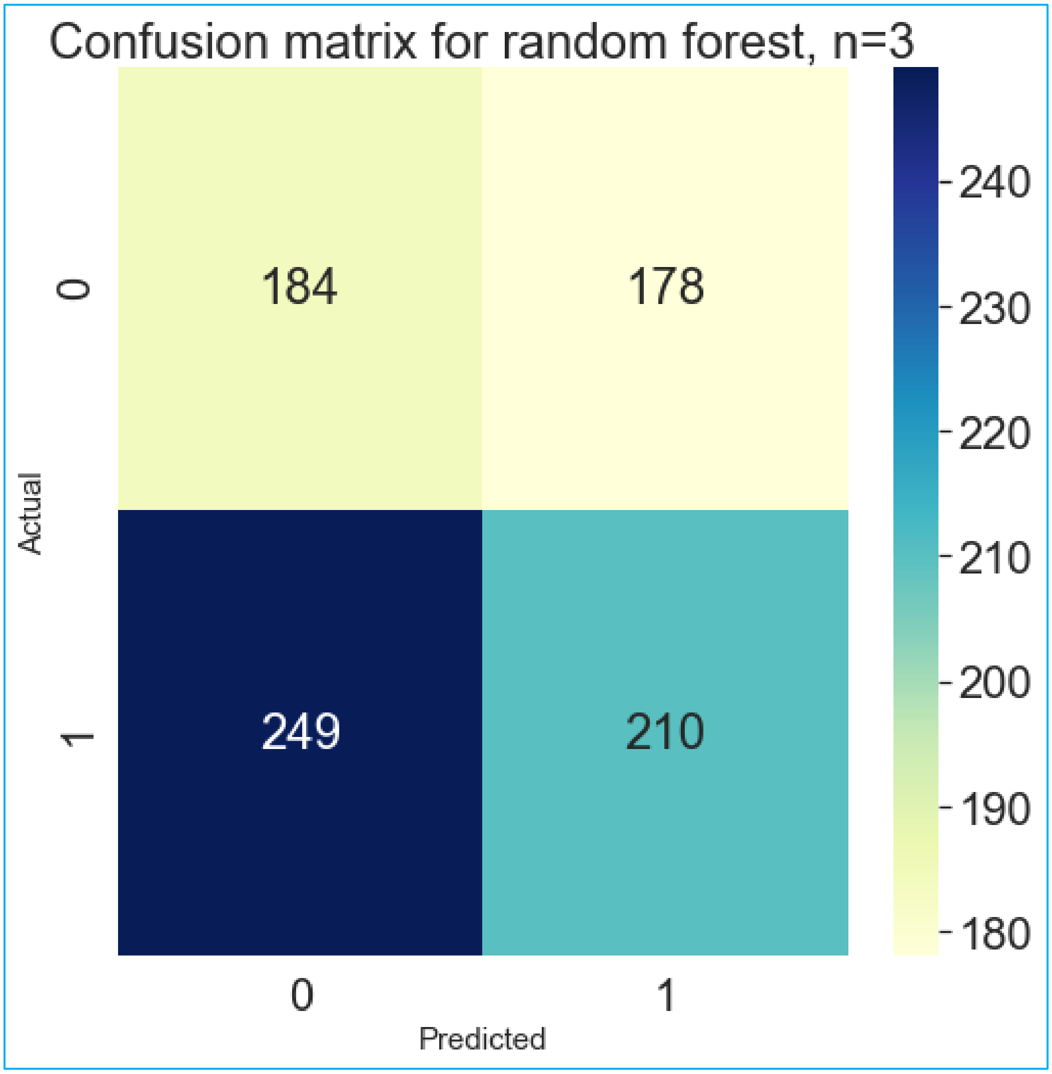
The top 5 negative words along with their coefficients for the Trigram Model are as follows:

| | Word | Coefficient |
|---|---|---|
| 349 | of climate change | -1.063907 |
| 98 | dark side of | -1.093729 |
| 350 | of human rights | -1.146323 |
| 690 | to try to | -1.161424 |
| 85 | child sex abuse | -1.273281 |

## iii. TRI-GRAM MODEL USING RANDOM FOREST:

```
In [444]: #random forest for n=3
          RFmodel3=RandomForestClassifier(n_estimators=800,criterion='entropy')
          RFmodel3=RFmodel3.fit(advancedtrain2,train['Label'])

In [445]: advancedtest2 = advancedvectorizer2.transform(testheadlines)
          preds5 = RFmodel3.predict(advancedtest2)
          acc5=accuracy_score(test['Label'], preds5)

In [446]: matrix=confusion_matrix(test['Label'],preds5)
          print(matrix)
          score=accuracy_score(test['Label'],preds5)
          print(score)
          report=classification_report(test['Label'],preds5)
          print(report)

[[191 171]
 [244 215]]
0.4945188794153471
              precision    recall  f1-score   support

           0       0.44      0.53      0.48       362
           1       0.56      0.47      0.51       459

    accuracy                           0.49       821
   macro avg       0.50      0.50      0.49       821
weighted avg       0.51      0.49      0.50       821
```

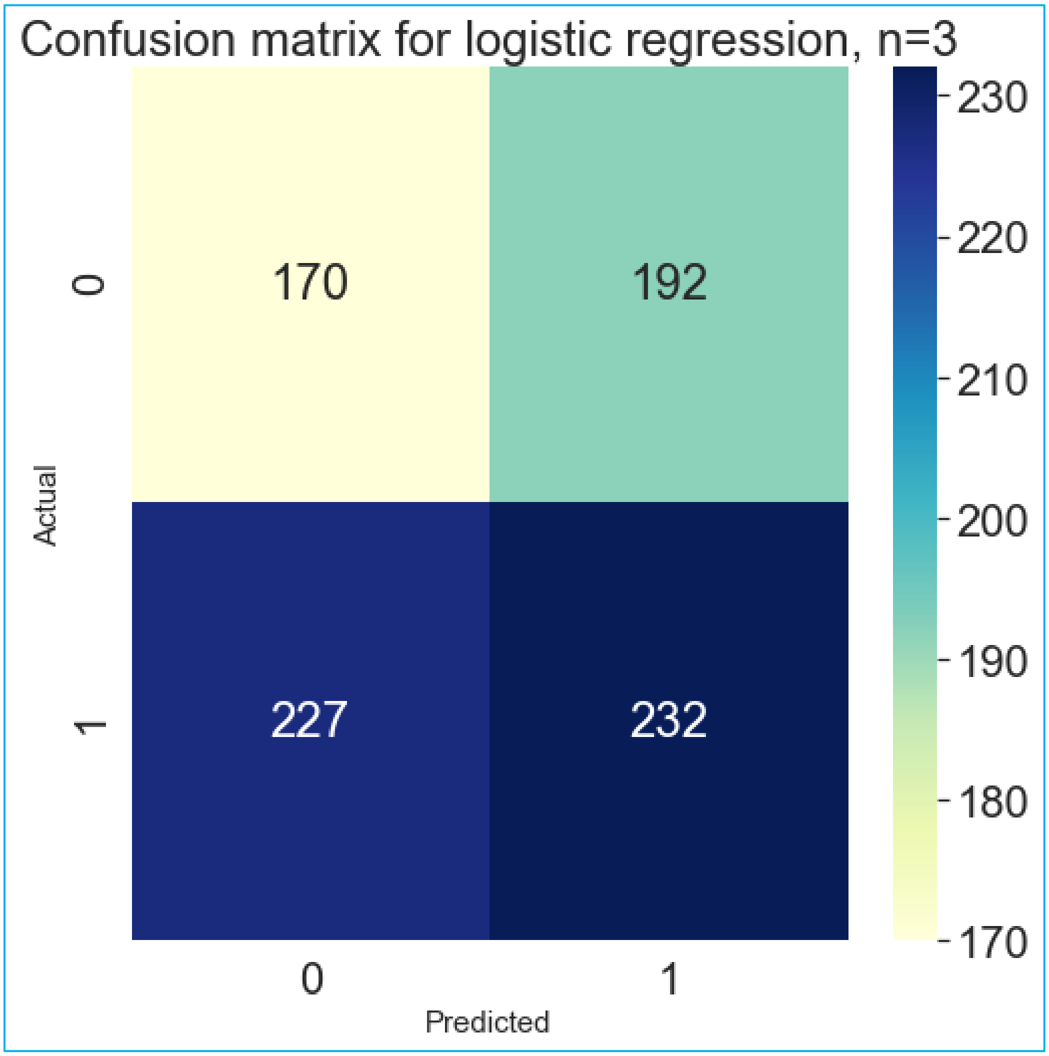
### ❖ TRI-GRAM MODEL RANDOM FOREST CONFUSION MATRIX:



## iv. TRI-GRAM MODEL USING LOGISTIC REGRESSION:

```
In [447]: #logistic regression for n=3
          LRmodel3 = LogisticRegression(max_iter=1000)
          LRmodel3 = LRmodel3.fit(advancedtrain2, train["Label"])

In [448]: advancedtest2 = advancedvectorizer2.transform(testheadlines)
          preds6 = LRmodel3.predict(advancedtest2)
          acc6=accuracy_score(test['Label'], preds6)

In [449]: matrix=confusion_matrix(test['Label'],preds6)
          print(matrix)
          score=accuracy_score(test['Label'],preds6)
          print(score)
          report=classification_report(test['Label'],preds6)
          print(report)

          [[170 192]
           [227 232]]
          0.48964677222898906
                        precision    recall  f1-score   support

                     0       0.43      0.47      0.45       362
                     1       0.55      0.51      0.53       459

              accuracy                           0.49       821
             macro avg       0.49      0.49      0.49       821
          weighted avg       0.49      0.49      0.49       821
```

❖ **TRI-GRAM MODEL LOGISTIC REGRESSION CONFUSION MATRIX:**



## 12. RESULT ANALYSIS:

After the comparative analysis of different N-gram models we find the following results:

v. **UNI-GRAM MODEL:**
The Random Forest Classifier gives an accuracy of 52.86% while the Logistic Regression Classifier gives an accuracy of 51.64%.

vi. **BI-GRAM MODEL:**
The Random Forest Classifier gives an accuracy of 50.66% while the Logistic Regression Classifier also gives the same accuracy of 50.66%.

vii. **TRI-GRAM MODEL:**
The Random Forest Classifier gives an accuracy of 49.45% while the Logistic Regression Classifier gives an accuracy of 48.96%.

## 13. CONCLUSION:

The accuracy of the model is the lowest for the trigram model and is the highest for unigram model. Hence, for the prediction of stock indices from the news headlines, using the random forest classifier of the unigram model, would give anyone the highest possibility of earning a profit if invested on that particular company.

## 14. REFERENCES


- Daskalopoulos, V. (2003). Stock price prediction from natural language understanding of news headlines. *TRIALS*, *1*(878), 0-988183.
- Falinouss, P. (2007). Stock trend prediction using news articles: a text mining approach.
- Kirange, M. D., & Deshmukh, R. R. (2016). Sentiment Analysis of News Headlines for Stock Price Prediction. *COMPUSOFT: An International Journal of Advanced Computer Technology*, *5*(3).
- Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., & Ngo, D. C. L. (2015). Text mining of news-headlines for FOREX market prediction: A Multi-layer Dimension Reduction Algorithm with semantics and sentiment. *Expert Systems with Applications*, *42*(1), 306-324.
- Oncharoen, P., & Vateekul, P. (2018, August). Deep learning for stock market prediction using event embedding and technical indicators. In *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)* (pp. 19-24). IEEE.
- *https://hackernoon.com/predict-stock-market-with-daily-top-news-8c8db25bef8d*
- *https://software.intel.com/content/www/us/en/develop/blogs/stock-predictions-through-news-sentiment-analysis.html*
- *https://www.essaysauce.com/finance-essays/news-headlines-sentiment-analysis-predict-stock-market-trends/*
- *https://www.sciencedirect.com/science/article/pii/S0970389619301569*
- *https://medium.com/@liuvivian/sentiment-of-news-headlines-as-a-directional-signal-of-stock-price-movement-321466b876d8*
- *https://towardsdatascience.com/sentiment-analysis-of-stocks-from-financial-news-using-python-82ebdcefb638*