HOMEWORK 3

Group Details:
Mohit Juneja 2017067
Preyansh Rastogi 2017176
Aarish Chhabra 2017212

Area Chosen: Brijpuri, Durgapuri

Twitter Handles: "@DelhiPolice ", "@CPDelhi ", "@DCPSouthDelhi ", "@DCPEastDelhi ", "@DCPSEastDelhi ", "@DCPNEastDelhi ", "@DcpNorthDelhi ", "@DCPCentralDelhi ", "@DCPNewDelhi "

a) Tweets collected using the above areas and twitter handles.
Total Unique Tweets: 107

i) All unrelated tweets (not related to riots) are removed.

Pre-processing tweets: Information extracted directly from tweets for further processing - ***tweet_date, tweet_username, tweet_replies, tweet_retweets, tweet_favorites, tweet_text, tweet_location, tweet_id, tweet_permalink.***

Hashtags and User mentions extracted using regular expressions.

```
for tweet_text in tweets_text :
    a = re.findall(r"\B(\#[a-zA-Z]+\b)",tweet_text)
    print(*a)

print ('Usermentions')
for tweet_text in tweets_text :
    a = re.findall(r"\B(\@[a-zA-Z]+\b)",tweet_text)
    print(*a)
```

Person Names and addresses / Locations are extracted using python 'nltk' library.
We have extracted the named entities in each of the tweets, person names and the locations are among the named entities after which we manually selected the person names and locations. We extracted the named entities by tagging of the tokenized tweet text, followed by chunking of the tags, the obtained chunks are then checked in the nltk Tree class to get the named entities.

```
tweets_text = load()
namedEntities = {}

for tweet_text in tweets_text:
    tokens = nltk.word_tokenize(tweet_text)
    tags = nltk.pos_tag(tokens)
    chunk = nltk.ne_chunk(tags)
    NE = [ " ".join(w for w,t in ele) for ele in chunk if isinstance(ele, nltk.Tree)]
    for entity in NE :
        if entity not in namedEntities :
            namedEntities[entity] = 0
        namedEntities[entity] += 1

list_ = []
for entity in namedEntities :
    list_.append((entity,namedEntities[entity]))
```

No contact numbers/vehicle numbers found from tweets.

```
print ('Mobile Numbers')
for tweet_text in tweets_text :
    a = re.findall(r"^((\+){1}91){1}[1-9]{1}[0-9]{9}$",tweet_text)
    print(*a)
```

For Vehicle Number:

```
pattern = "([A-Z]{2}[ -]*[0-9]{1,2}(?:[A-Z])?(?:[ -]*[A-Z]*)?[ -]*[0-9]{4})"
for tweet in tweets_text:
    a = re.findall(pattern,tweet)
    print(*a)
```

ii) Tweets annotated manually into categories: **action-related, chaos related, help-related, rumors related.**

Features Used for Manual Annotation:

Features:
Verified, Profile location (Yes/No) , Profile Name (Yes/No), Profile Picture(yes/No), Account joining date, Fav+retweet+reply count, duplicate tweets (Yes/No), Event location == profile location, profile description (yes/no) , presence of phrase, "I see"

And

Table 2: Description of new features

Category	Feature Name
Source Credibility	Is trusted/satirical news account
	Has trusted/satirical news url
	Profile has url from top domains
	Client application name
Source Identity	Profile has person name
	Profile has location
	Profile includes profession information
Source Diversity	Has multiple news/non-news urls after dedup
	Deduped tweets' text is dissimilar
Source Location & Witness	If tweet location matches event location
	If profile location matches event location
	Has witness phrases, i.e. "I see" and "I hear"
Msg. Belief	Is support, negation, question or neutrality
Event Propagation	Event Topic
	Retweet, mention, hashtag h-index
	Max reply/retweet graph[4] size/depth

All tweets stored in a CSV file along with classifications made.

b)

i) As the number of tweets collected is very less in the locality chosen, an external dataset is used with annotated tweets to increase the size of the dataset.

Dataset link - Link

This dataset is used to train the model and our annotated dataset is used as a test set.

Features Used for prediction: **['verified', 'profile_location', 'profile_name', 'profile_image', 'profile_desc', 'fav_count', 'retweet_count', 'text', 'media',]**

Textual features(name, location, description, tweet_text) are converted to float values using Sentiment Analysis. **VADER Sentiment Analysis** library of python is used for this purpose.

Boolean features (verified, media) are converted to integral values.

Model: Multinomial Naive Bayes classifier is used as it works well for discrete features.

'**predict_proba()'** function is used to calculate the rumor spread for any tweet.

```
def rumour_spread(id):
    id = str(id)
    cur_feature = featureSet_test[ids[id]]
    print(model.predict_proba(cur_feature.reshape(1, -1)))
```

The function returns the probability of a tweet being a rumor/non-rumor.

ii) Evaluation Metrics:

1) Accuracy

   Accuracy on Test Set:
   0.5794392523364486

   Accuracy = (Number of samples correctly detected) / (Total number of samples)

2) Confusion Matrix

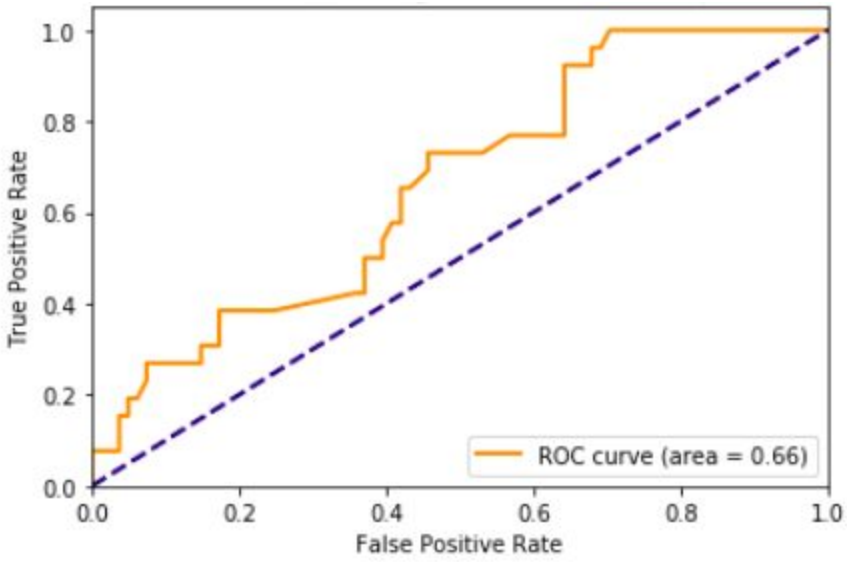
   Confusion Matrix for the model:
   [[51 30]
    [15 11]]

   The confusion matrix provides a good evaluation metric. The 4 values in the confusion matrix denote:

(0,0) : True Negative (Non-Rumour detected as Non-Rumour) -- 51
(0,1) : False Negative (Non-Rumour detected as Rumour) -- 30
(1,0) : False Positive (Rumour detected as Non-Rumour) -- 15
(1,1) : True Positive (Rumour detected as Rumour) -- 11

Clearly, **True Positives + True Negatives > False Positives + False Negatives**, thus signifying that the model is obviously better than any random guess and the same is also justified by the accuracy mentioned above.

3) ROC Curve



The ROC curve is the curve between True Positive Rate Vs False Positive Rate. For a model to be better than a random guess, the curve must be above the **y=x** line. The curve above satisfies the same and even the area of the ROC curve is 0.66 > 0.50.

Hence, visualizing all the metrics, the model is a decent model much better than a random guess.

iii) **Examples where the model did well:**

Tweet Url	Actual Label	Predicted Label	Reasons
URL	Rumour	Rumour	Not verified, no location and location in profile, no proper English and no media attached in the tweet.

			No retweet and fav on the tweet.
URL	Rumour	Rumour	Profile description with negative sentiment(contain word ‘fight’), tweet in Hindi - not formal, not verified and no location in profile, no media in the tweet.
URL	Non-Rumour	Non-Rumour	Verified profile of LG Delhi with a positive profile description. A good number of retweets and favorites on the tweet. Image source in the tweet.
URL	Non-Rumour	Non-Rumour	The user’s profile has location, name as well as image.
URL	Rumour	Rumour	The tweet has no media. The account is not verified. The description has some non-English words written in English, so the sentiments come out to be neutral rather than positive.

Examples where the model failed:

Tweet Url	Actual Label	Predicted Label	Reasons
URL	Non-Rumour	Rumour	The tweet has no media. The account is not verified. The description has some non-English words written in English, so the sentiments come out to be neutral rather than positive.
URL	Rumour	Non-Rumour	The user’s profile has location, name as well as image. The text has no biased or negative sentiments towards anyone. So, it has been labeled as a Non-Rumour.
URL	Rumour	Non-Rumour	The user’s profile has location, description, name as well as image. Description, as well as the text, have no biased or negative sentiments towards anyone. So, it has been labeled as a Non-Rumour.
URL	Non-Rumour	Rumour	The tweet has no likes and retweets. So, therefore the model predicted it as a Rumour.
URL	Non-Rumour	Rumour	It could be predicted as Rumour, since the account was created recently (December 2019).

iv) Multiple ways to improve the model:

1) Image/video analysis could be performed on the media attached to the tweets to make better predictions.
2) Features could include profile information about followers/following of the user. Fake profiles could be identified with this.
3) Twitter account date can also be used as a feature to determine whether the profile has been made just for rumor spread.

v) Our model could be used to make the predictions given any tweet. Other features mentioned in part (iv) may also be added to the classifier.
We have made a simple app(minimal UI) using the 'Tkinter' library and the trained classifier.

Steps to use the application:

1) Run 'app.py' file
2) Pass the URL of the tweet in the label text field and click the search button.
3) Prediction from the model will be returned.

NOTE: "model" named pickle file must be stored in the same folder as "app.py".

Steps to develop the application:

1) Given the URL of the tweet, extract all the features from the tweet required to make the predictions.

2) Convert all the textual features into numerical values using sentiment analysis library etc.

```
def feature_list(url) :
    # url will be of the form https://twitter.com/handle/status/tweet_id

    handle = url.split('/')[-3]
    id_ = int(url.split('/')[-1])
    tweet = api.get_status(id_,tweet_mode='extended')
    row_new = [0 for i in range(9)]
    check = 0
    user = api.get_user(screen_name=handle)
    if user.verified:
        row_new[0]=1

    if user.location is not None:
        row_new[1]=1

    if user.name is not None or user.name!="":
        row_new[2]=1

    if "profile_image_url_https" in user.profile_image_url_https!="":
        row_new[3]=1

    row_new[4] = get_sentiment(user.description,"en")
    row_new[5] = tweet.favorite_count
    row_new[6] = tweet.retweet_count
    row_new[7] = get_sentiment(tweet.full_text,"en")

    featureSet_test = np.array([row_new])
    featureSet_test = np.asarray(featureSet_test, dtype='float64')

    return featureSet_test
```

3) Pass the features to the model to get the predictions.

```
def solve(url = 'https://twitter.com/PettyPraxis/status/1232793544860979201')
    model = getmodel()
    featureSet_test = feature_list(url)
    predict = model.predict(featureSet_test)
    if (predict == '0') :
        return 'Not Rumour'
    else :
        return 'Rumour'
```

4) Make a simple GUI that takes the URL of the tweet as an input and provides the label as output.