CS 579: Online Social Network Analysis Project II

Fake News Classification Report

Group Members:

Gonuguntla Venkata Sai Goutham - A20450688 - vgonuguntla@hawk.iit.edu

Rishikesh Jangam – A20448930 – rjangam@hawk.iit.edu

Introdcution:

"Fake News" is a term used to represent made-up news or propaganda comprising misinformation communicated through social media. Spreading of fake news is one of the greatest threats to an individual or a country. With the increasing popularity and outreach of social media, such fake or misinformation could spread faster than ever before, reach a bigger audience, and influence public opinion. Fake news is increasingly being shared via social media platforms like Facebook, Twitter, Instagram and many more platforms. These platforms offer a setting for the general population to share their opinions and views to the world. Research that studied the spread of fake news concluded that tweets containing false information reach people on Twitter six times faster than truthful tweets. Fake news come in many forms including: unintentional errors committed by news aggregators, outright false stories or the stories which developed to mislead and influence readers' opinion. While fake news may have multiple forms, the effect that it can have on people, government and organizations may generally be negative since it differs from facts. The very task of defining the fake news is a challenge. Artificial Intelligence and Natural Language Processing tools offer great promise for researchers to build systems which could detect fake news. However, detecting fake news is a challenging task to accomplish as it requires models to summarize the news and compare it to the actual news to classify it as fake.

The goal of this project is to identify whether a headline or an article is "Fake" or not. The project involves comparing the headlines with a body of text from a news article to determine what relationship exists between two.

We need to classify into three categories:

- Agreed: B talks about the same fake news as A
- disagreed: B refutes the fake news in A.
- unrelated: B is unrelated to A.

In this project we have tried to address the problem of fake news by developing a classifier which can automatically detect the fake news accurately. A classifier can solve the problem with high accuracy might effectively be used either as a tool for human working to identify fake news.

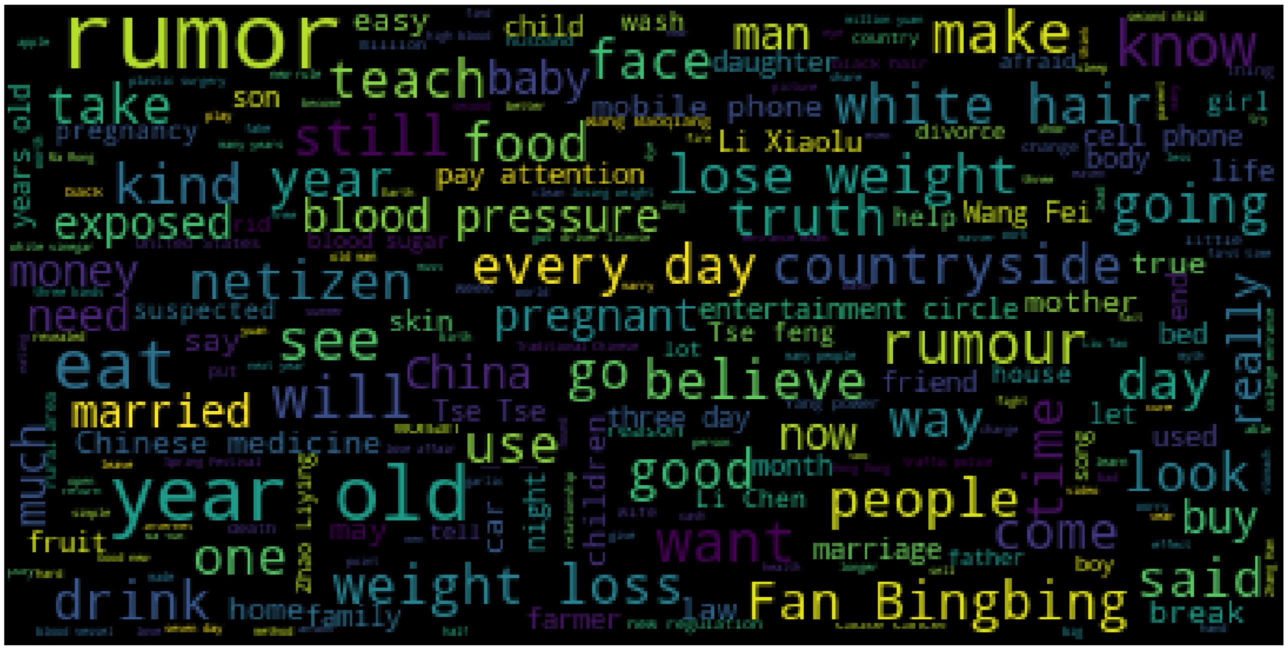
Understanding the Data:

In the train.csv we have title1 which is the fake news title and title2 is the news tile and label column tells about the relation between the title1 and title2 whether it agrees, disagrees or unrelated.

```
train_df.head()
```

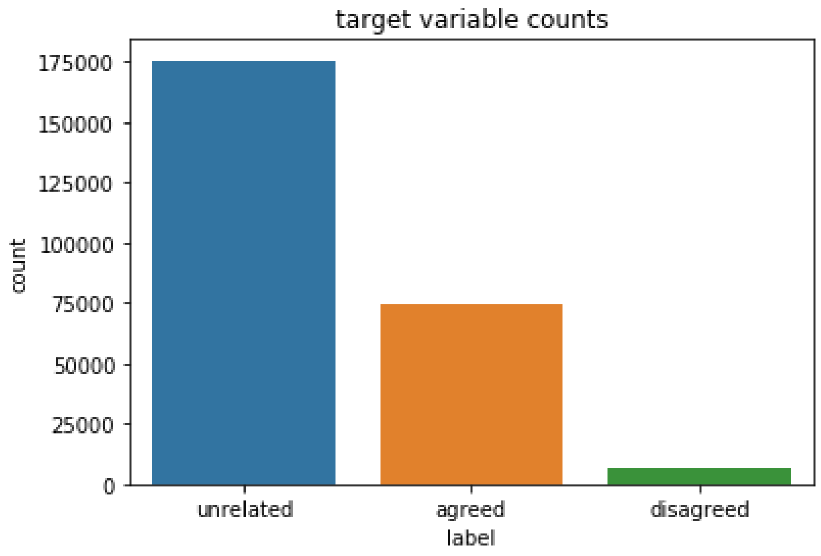
	id	tid1	tid2	title1_en	title2_en	label
0	195611	0	1	There are two new old-age insurance benefits f...	Police disprove "bird's nest congress each per...	unrelated
1	191474	2	3	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outstrips Hong Kong? Shenzhen S...	unrelated
2	25300	2	4	"If you do not come to Shenzhen, sooner or lat...	The GDP overtopped Hong Kong? Shenzhen clarifi...	unrelated
3	123757	2	8	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP overtakes Hong Kong? Bureau of ...	unrelated
4	141761	2	11	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outpaces Hong Kong? Defending R...	unrelated

We have 256442 rows in the train csv which is where hard to go and read each and every title and understand what the data is about. So, we have generated word cloud image on the two columns collectively.



The above image is the word cloud image. This image tells about word frequency or importance in the dataset. Bigger the size the more frequent word appears or more important. We can see some words like rumor, lose weight, country side and so many words which are having bigger size may be the articles in the dataset are based on these words.

Target Value label distribution:



The above chart shows the how the target variable category is distributed in the whole dataset, it seems the unrelated label rows are very high.

The rest of the report is organized as follows: Preprocessing and Feature Engineering describes the dataset, cleaning and making dataset for building the model, Model Description explains everything about how the model works and how the fake news is classified, and Results presents the accuracy of each model and which model is best for detecting fake news.

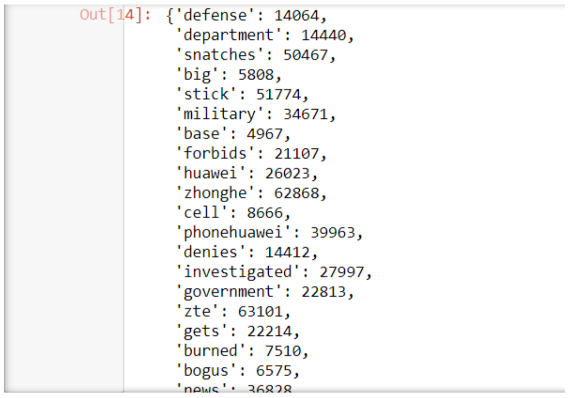
Data - Pre Processing and Feature Engineering:

As part of data preprocessing we have to remove the stop words are the commonly used words such as a, an, the, in and so on from the data. Also we considered removing the punctuations. Lemmatization of words is also performed so that the group of words are considered with a single word. For example, run will be lemmatized word for running, ran, run. Along with these transformations we have also ignored the numeric values but considered the alphanumeric and rest.

After doing the above transformations we have calculated the tf-idf vectorizer values and count vectorizer values in 2 different scenarios.

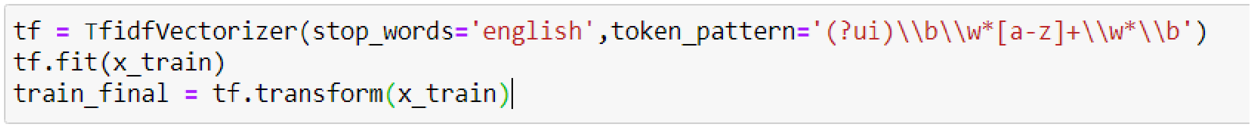
In the first scenario we have calculated the tf-idf vectorizer values and count vectorizer values on each column individually and combine them for model training but doing this we have experienced the memory issue and performance was not up to the expectation.

So, we tried different scenario where we have created a new column which is concatenation of title1 and title2 columns row-wise. On this new column we have calculated tf-idf vectorizer values and this image gives an idea of on what words the model is trying to calculate tf-idf values as you



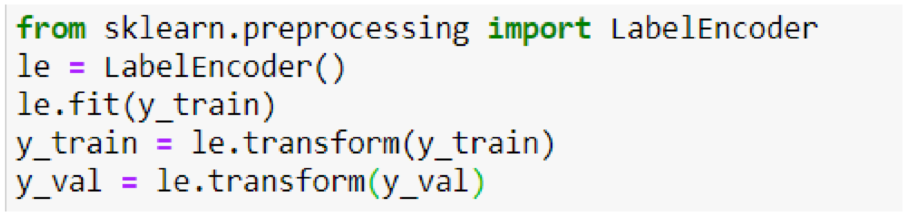
```
Out[14]: {'defense': 14064,
          'department': 14440,
          'snatches': 50467,
          'big': 5808,
          'stick': 51774,
          'military': 34671,
          'base': 4967,
          'forbids': 21107,
          'huawei': 26023,
          'zhonghe': 62868,
          'cell': 8666,
          'phonehuawei': 39963,
          'denies': 14412,
          'investigated': 27997,
          'government': 22813,
          'zte': 63101,
          'gets': 22214,
          'burned': 7510,
          'bogus': 6575,
          'news': 36828
```

can see there are no numeric words, puntuations or stop words in the vocabulary generated. Using this vocabulary tf-idf values are calculated for each row. This transformed tf-idf vectors are input values to train the model. The code for this tfidf vector is shown below.



```
tf = TfidfVectorizer(stop_words='english',token_pattern='(?ui)\\b\\w*[a-z]+\\w*\\b')
tf.fit(x_train)
train_final = tf.transform(x_train)
```

The target variable category has been encoded to 0,1,2 for agree,disagree and unrelated respectively. This can be done using labelEncoder() function.

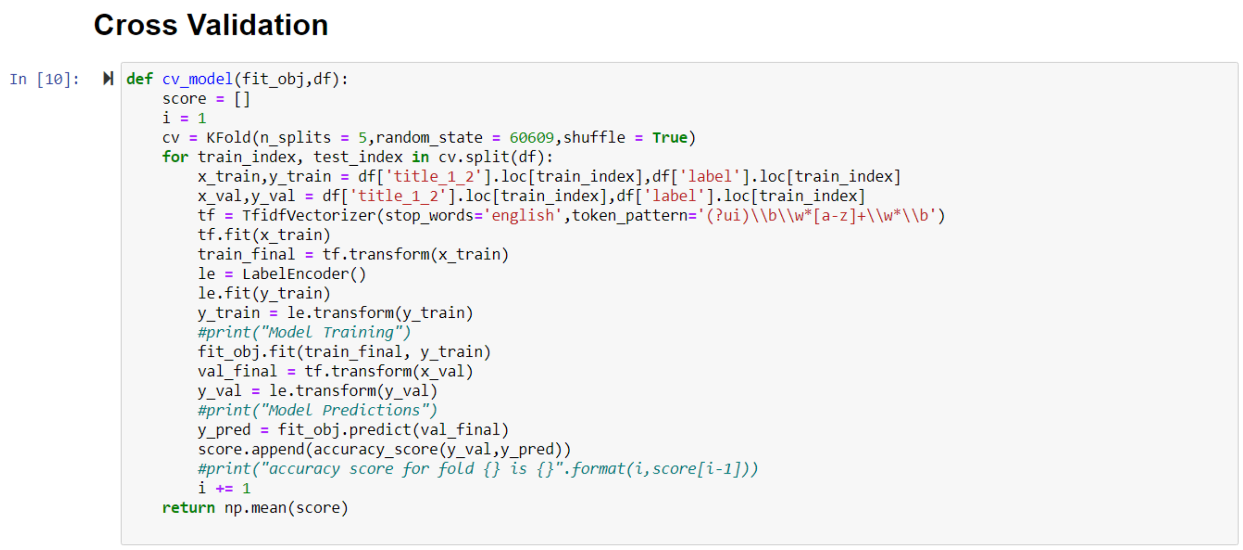


```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(y_train)
y_train = le.transform(y_train)
y_val = le.transform(y_val)
```

The above are the data pre processing and transformations performed on the training dataset which gave the best performance on the test dataset.

Model Description:

After performing the data pre-processing and transformatins on the training dataset. To build the model we performed 5-fold cross validation on different models to decide which model gives the best performance.

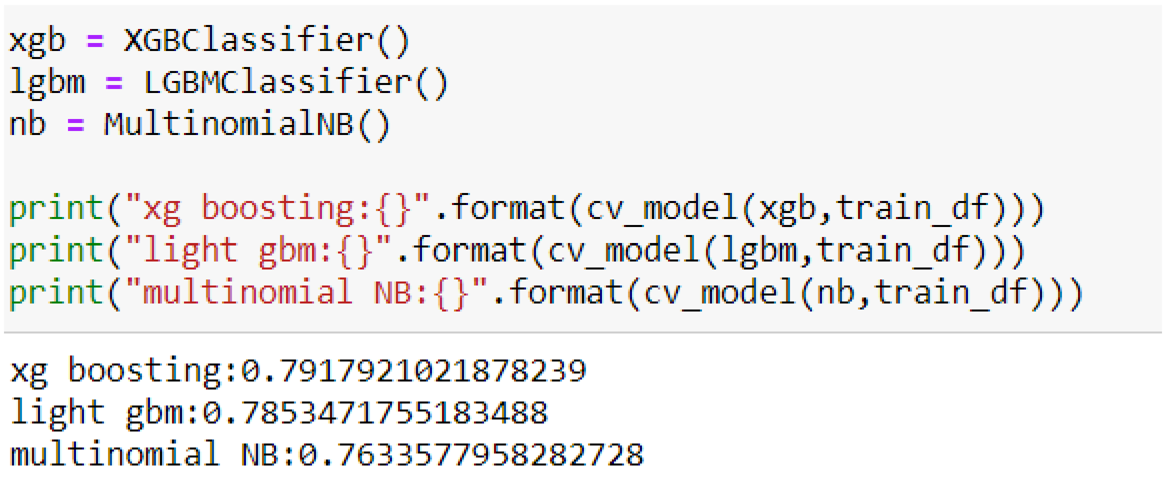


Cross Validation

```
In [10]:  def cv_model(fit_obj,df):
              score = []
              i = 1
              cv = KFold(n_splits = 5,random_state = 60609,shuffle = True)
              for train_index, test_index in cv.split(df):
                  x_train,y_train = df['title_1_2'].loc[train_index],df['label'].loc[train_index]
                  x_val,y_val = df['title_1_2'].loc[train_index],df['label'].loc[train_index]
                  tf = TfidfVectorizer(stop_words='english',token_pattern='(?ui)\\b\\w*[a-z]+\\w*\\b')
                  tf.fit(x_train)
                  train_final = tf.transform(x_train)
                  le = LabelEncoder()
                  le.fit(y_train)
                  y_train = le.transform(y_train)
                  #print("Model Training")
                  fit_obj.fit(train_final, y_train)
                  val_final = tf.transform(x_val)
                  y_val = le.transform(y_val)
                  #print("Model Predictions")
                  y_pred = fit_obj.predict(val_final)
                  score.append(accuracy_score(y_val,y_pred))
                  #print("accuracy score for fold {} is {}".format(i,score[i-1]))
                  i += 1
              return np.mean(score)
```

The above is the code for 5 fold cross validation, in this KFold function retruns the 5 different train, validation indices having random shuffling in the each set of data and random state is set to 60609 to replicate the same results any time.

After this, we are extracting the data based on train,validation indices given b KFold function and on the train data we fit tf-idf and transform the train, validation data on the train data. We encode the target variable to 0,1,2 based on the fit of y_train data. Once, the pre processing steps are performed the function takes the different model objects and fit the data and perform the predictions on the validation data for each fold. Once the predictions are done the accuracy score is caluclated. Finally, this function returns the mean of the accuracy score for 5- folds of the model object that was passed.



```
xgb = XGBClassifier()
lgbm = LGBMClassifier()
nb = MultinomialNB()

print("xg boosting:{}".format(cv_model(xgb,train_df)))
print("light gbm:{}".format(cv_model(lgbm,train_df)))
print("multinomial NB:{}".format(cv_model(nb,train_df)))
```

```
xg boosting:0.7917921021878239
light gbm:0.7853471755183488
multinomial NB:0.7633577958282728
```

Now, We have tried different models such as xgboost classifier, multinomial naïve bayes classifier and light gradient boost classifier machine learning models as shown above.

These machine learning model instances are created and passed through the cv function and we can see the mean accuracy score for each model with 0.79, 0.78,0.76. We can see that xgboost model works out best in the cross validation process.

We have then split the training data to train, validation splits of 70,30 percent of training and built the machine learning models.

Results:

Model	**Score**
XGBoost	0.767
Light Gradient Boosting	**0.772**
Multinomial Naïve Bayes	0.747

The above are the results based on train, test split of 70,30 on different models. Light Gradient boosting has high accuracy score of 0.772 so the final model predictions on test dataset are done using light gradient boosting machine learning model.

Project Justification:

- Data Pre-processing and Feature Engineering - Gonuguntla Venkata Sai Goutham
- Model Description -
  - XGBoost - Gonuguntla Venkata Sai Goutham
  - Light Gradient Boosting - Gonuguntla Venkata Sai Goutham
  - Multinomial Naïve Bayes – Rishikesh Jangam
- Results - Gonuguntla Venkata Sai Goutham, Rishikesh Jangam
- Report - Gonuguntla Venkata Sai Goutham, Rishikesh Jangam
- Presentation - Gonuguntla Venkata Sai Goutham, Rishikesh Jangam

References:

- Light GBM - Python-package Introduction — LightGBM 3.1.0.99 documentation
- Xgboost - XGBoost Documentation — xgboost 1.3.0-SNAPSHOT documentation
- Multinomial Naïve Bayes - sklearn.naive_bayes.MultinomialNB — scikit-learn 0.23.2 documentation (scikit-learn.org)
- Tf idf - sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 0.23.2 documentation (scikit-learn.org)
- Fake News Challenge
- sklearn.model_selection.KFold — scikit-learn 0.23.2 documentation (scikit-learn.org)