Cryptography Project (CS352)

May 15, 2020

Classification of Various Fake News Articles

Name: Dheeraj Chaudhary

Roll: 17BCS009

1 Description of Dataset

I have taken a dataset of Fake News Articles, which contains Fake news kept in rows, with various relevent columns, which we'll see later in the Analysis. The dataset contains various articles of fake News. This dataset was collected from realworld sources; The fake news articles were collected from different sources like, from unreliable websites that were flagged by Politifact (a fact-checking organization in the USA) and Wikipedia. The dataset contains different types of articles on different topics, however, the majority of articles focus on political and World news topics.

> **PS**: The dataset contains Four columns, First is **title** which have titles of the article, second is **text** which have content/story of the article, Third is **subject**, to what category the article belong and the last column is **date** of posting of the article.

Name: ISOT Fake News Dataset

Provider: External Data Source

Host: University of Victoria

1.0.1 Using this dataset, I'll apply various Machine Learning models to get the best possible one, which can classify the fake news articles into various categories like, whether it belongs to *Political news* or *government news* etc.

1.1 Let's have a look at the model used and the accuracy achieved in each, along with some important terms used

- **Accuracy: (True Positive + True Negative) / Total Population**

- Accuracy is a ratio of correctly predicted observation to the total observations. Accuracy is the most intuitive performance measure.
  - True Positive: The number of correct predictions that the occurrence is positive
  - True Negative: The number of correct predictions that the occurrence is negative
- **F1-Score: (2 x Precision x Recall) / (Precision + Recall**
  - F1-Score is the weighted average of Precision and Recall used in all types of classification algorithms. Therefore, this score takes both false positives and false negatives into account. F1-Score is usually more useful than accuracy, especially if you have an uneven class distribution.
- **Precision:** When a positive value is predicted, how often is the prediction correct?
- **Recall:** When the actual value is positive, how often is the prediction correct?

Model Name	Accuracy	Precision	F1-Score
Multinomial Classifier	0.55	0.50	0.52
XGB Classifier	0.62	0.59	0.60
Logistic Regression	0.64	0.62	0.62
Support Vector Machine	0.70	0.62	0.63
Gradient Boosting Classifier	0.70	0.66	0.65
Stochastic Gradient Descent	0.70	0.66	0.65

1.1.1 We can see last three models are giving same accuracy of 70%, which is best which I've achieved for the given dataset after applicying on various other techniques too

1.2 The Analysis is divided into three sections:

1.2.1 Analysing Data

1.2.2 Preprocessing on Data

1.2.3 Implementing Machine Learning on the final Data

1.2.4 Dataset Citation

2 Analysing Data

2.1 Now, I will discuss the methods used for above mentioned model and Analyse the dataset

[140]:
```
## importing the reuqired libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.pipeline import Pipeline
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
import nltk
from nltk.tokenize import RegexpTokenizer
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
import seaborn as sns; sns.set()
```

2.2 Load the dataset and print starting rows, along with describe the dataset in the next cell

[141]:
```
df = pd.read_csv('/home/dheeraj/my_projects/my_project_env/practice/
↪6th_sem_Academics/Cryptography/Project/Fake.csv')
df.head()
```

[141]:
```
                                               title  \
0   Donald Trump Sends Out Embarrassing New Year’...
1   Drunk Bragging Trump Staffer Started Russian ...
2   Sheriff David Clarke Becomes An Internet Joke...
3   Trump Is So Obsessed He Even Has Obama’s Name...
4   Pope Francis Just Called Out Donald Trump Dur...

                                                text subject  \
0  Donald Trump just couldn t wish all Americans ...    News
1  House Intelligence Committee Chairman Devin Nu...    News
2  On Friday, it was revealed that former Milwauk...    News
3  On Christmas day, Donald Trump announced that ...    News
4  Pope Francis used his annual Christmas Day mes...    News

                date
0  December 31, 2017
1  December 31, 2017
2  December 30, 2017
3  December 29, 2017
4  December 25, 2017
```

[142]:
```
df.describe()
```

[142]:
```
                                                    title   text subject  \
count                                               23481  23481   23481
unique                                              17903  17455       6
top     MEDIA IGNORES Time That Bill Clinton FIRED His...             News
```

```
freq                                                               6     626     9050

                 date
count           23481
unique           1681
top      May 10, 2017
freq               46
```

2.3 Printing all the unique category present in the dataset

2.3.1 we can see all the 6 categories and number of articles belong to each corresponding category

[143]:
```
category = df['subject'].value_counts()
category
# plt.plot(category)
```

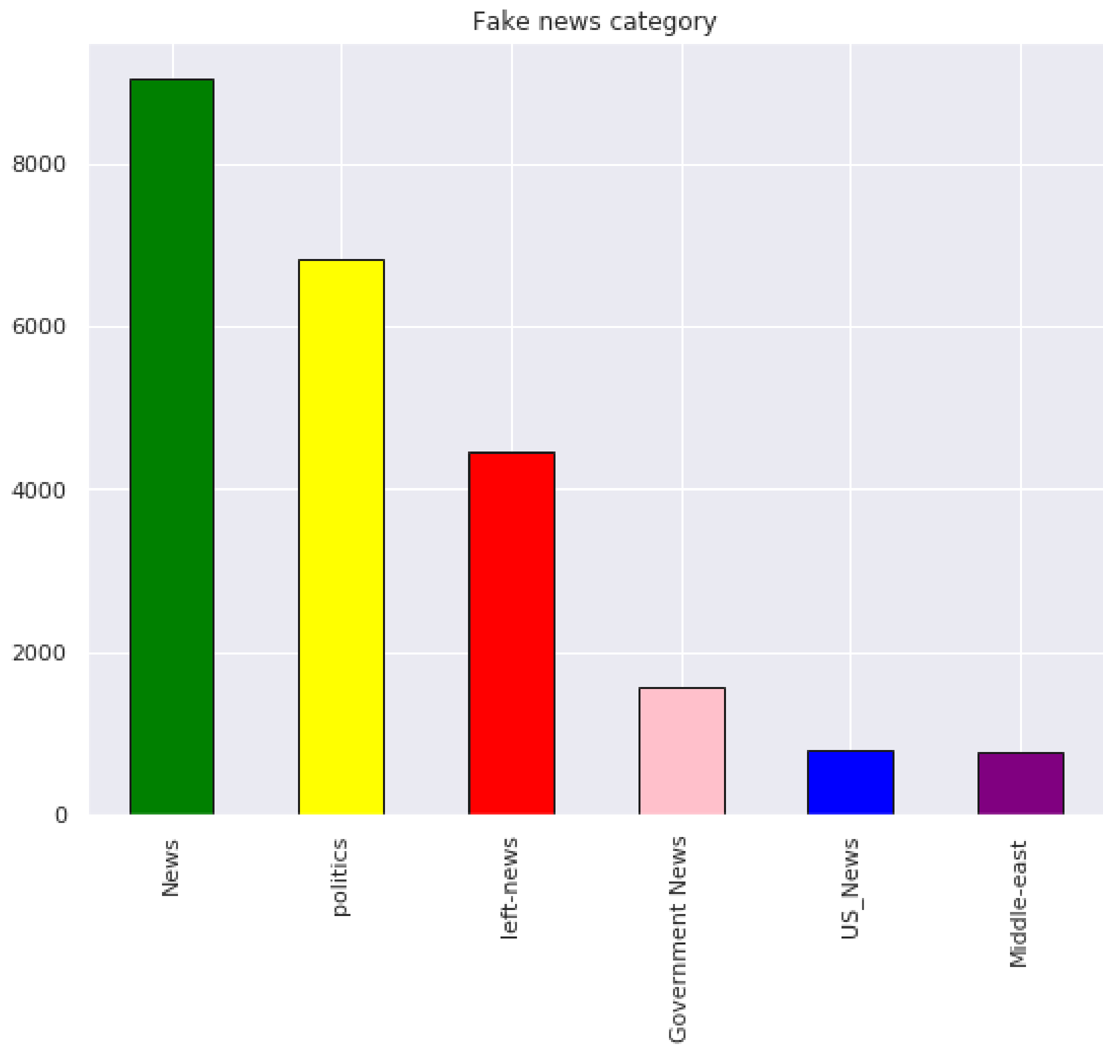
[143]:
```
News               9050
politics           6841
left-news          4459
Government News    1570
US_News             783
Middle-east         778
Name: subject, dtype: int64
```

2.4 Plotting Bar graph for categories of news articles

[144]:
```
colors = ['green', 'yellow', 'red', 'pink', 'blue', 'purple' ]
df['subject'].value_counts().plot(kind='bar',figsize=(9,␣
↪7),edgecolor='k',title="Fake news category", colors = colors)
# plt.savefig('Sentiment_bar_plot.png', dpi=100)
```

```
/home/dheeraj/my_projects/my_project_env/lib/python3.6/site-
packages/pandas/plotting/_matplotlib/core.py:203: UserWarning: 'colors' is being
deprecated. Please use 'color'instead of 'colors'
  "'colors' is being deprecated. Please use 'color'"
```

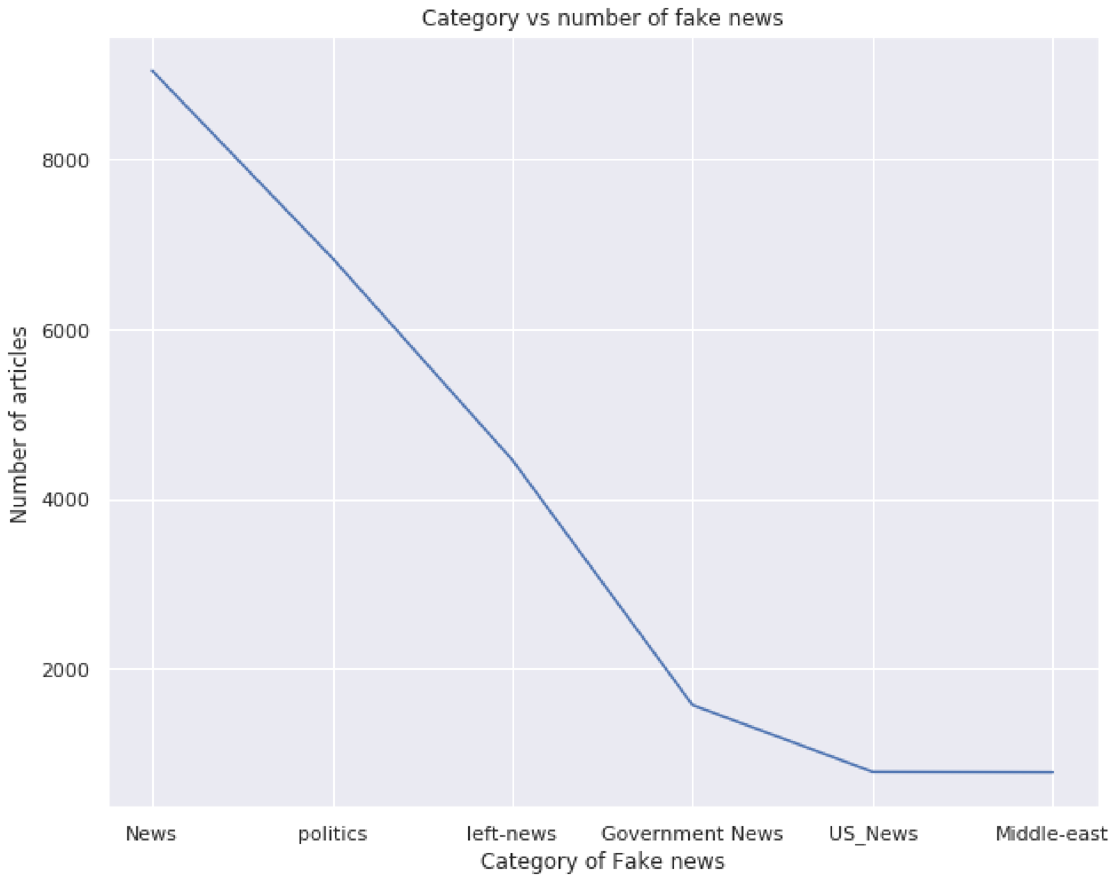
[144]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7eff70ce2908>
```



2.5 Ploting the plots for subject vs number of fake news

[145]:
```
fig = plt.figure(figsize=(10,8))
category = df['subject'].value_counts()
# category
plt.plot(category)
plt.xlabel("Category of Fake news")
plt.ylabel("Number of articles")
plt.title('Category vs number of fake news')
```

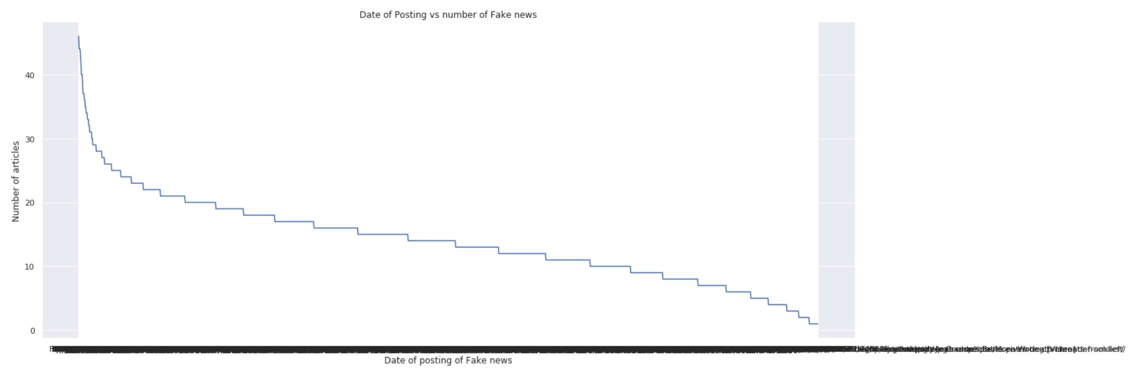
[145]: Text(0.5, 1.0, 'Category vs number of fake news')



2.6 Ploting the plots for dates vs number of fake news starting from 2017 and back to 2015

[146]:
```
fig = plt.figure(figsize=(20,8))
dates = df['date'].value_counts()
# dates
plt.plot(dates)
plt.xlabel("Date of posting of Fake news")
plt.ylabel("Number of articles")
plt.title('Date of Posting vs number of Fake news')
```

[146]: Text(0.5, 1.0, 'Date of Posting vs number of Fake news')



3 Preprocessing on Data

Since, we can see number of articles belongs to US_News and Middle-east are very less comapred to other category. So, we need to discard it from our dataset. > To do this, we'll make a new column in the datset as, label. And assign all those articles a unique number which are not belong to US_News and Middle-east. So finally we'll see, in the label column that all corresponding value of these two discarded category will be *NA*.

[147]:
```
df["label"] = df['subject'].map({'News':0,
'politics' : 1,
'left-news': 2,
'Government News' : 3})
```

3.1 Here finally we'll take the column which are required for our Machine Learning model.

3.1.1 Taking the article text column which contains story of the Fake news and label column which we just have added in the dataset.

We'll also drop all those rows which have *NA* value in the label column, which refers that we are removing article belongs to US_News and Middle-east news category.

[148]:
```
model_data = pd.DataFrame()
model_data['text'] = df['text']
model_data['label'] = df['label']
model_data = model_data.dropna()
model_data['text']
```

[148]:
```
0        Donald Trump just couldn t wish all Americans ...
1        House Intelligence Committee Chairman Devin Nu...
2        On Friday, it was revealed that former Milwauk...
3        On Christmas day, Donald Trump announced that ...
4        Pope Francis used his annual Christmas Day mes...
```

```
                              ...
21915    Don t you just love an entitled IRS lawyer who...
21916    This is a sad commentary on a generation who h...
21917    Yeah that whole  taking up arms  thing seems t...
21918    In case you missed it Sen. Harry Reid (R-NV), ...
21919    The irony here isn t lost on us. Hillary is be...
Name: text, Length: 21920, dtype: object
```

3.2 Some preprocessing in the *text* column so that it helps our model to understand the category of the news.

[149]:
```
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer()
text_counts= cv.fit_transform(model_data['text'])
```

4 Implementing Machine Learning on the final Data

4.1 Splitting the final dataset into train and test sets of 70:30, and printing the corresponding values

4.2 Here x is story/text column, and y is labels corresponding to each story category

[150]:
```
X_train, X_test, y_train, y_test = train_test_split(
    text_counts, model_data['label'], test_size=0.30, shuffle=True,␣
 ↪random_state=101)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(15344, 89272)
(6576, 89272)
(15344,)
(6576,)
```

4.3 Prediction using Multinomial model

[151]:
```
multinomial_model = MultinomialNB().fit(X_train, y_train)
multinomial_model_predicted= multinomial_model.predict(X_test)
```

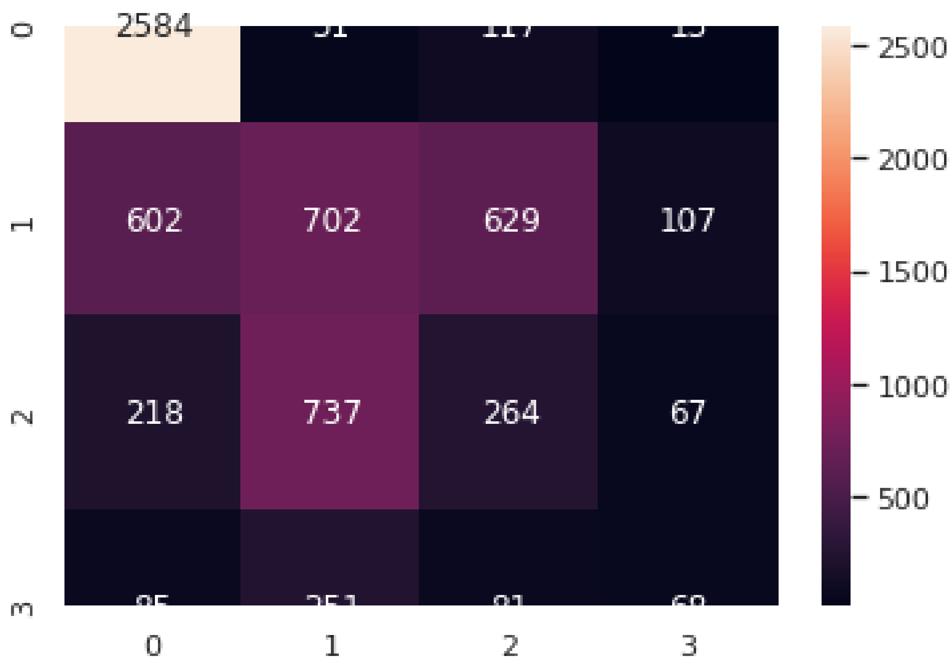
[152]:
```
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test,␣
 ↪multinomial_model_predicted))
```

```
MultinomialNB Accuracy: 0.5501824817518248
```

[153]:
```
print("confusion_matrix",confusion_matrix(y_test,multinomial_model_predicted))
```

```
confusion_matrix [[2584   51  117   13]
 [ 602  702  629  107]
 [ 218  737  264   67]
 [  85  251   81   68]]
```

[154]:
```python
import numpy as np; np.random.seed(0)
import seaborn as sns; sns.set()
uniform_data = confusion_matrix(y_test,multinomial_model_predicted)
ax = sns.heatmap(uniform_data, annot=True, fmt="d")
```



[155]:
```python
print("classification_report")
print(classification_report(y_test,multinomial_model_predicted))
```

```
classification_report
              precision    recall  f1-score   support

         0.0       0.74      0.93      0.83      2765
         1.0       0.40      0.34      0.37      2040
         2.0       0.24      0.21      0.22      1286
         3.0       0.27      0.14      0.18       485

    accuracy                           0.55      6576
   macro avg       0.41      0.41      0.40      6576
weighted avg       0.50      0.55      0.52      6576
```

4.4 Prediction using XGBClassifier model

[156]:
```
xgb_model = XGBClassifier()
xgb_model.fit(X_train, y_train)
```

[156]:
```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method=None, validate_parameters=False, verbosity=None)
```

[157]:
```
xgb_model_predicted = xgb_model.predict(X_test)
```

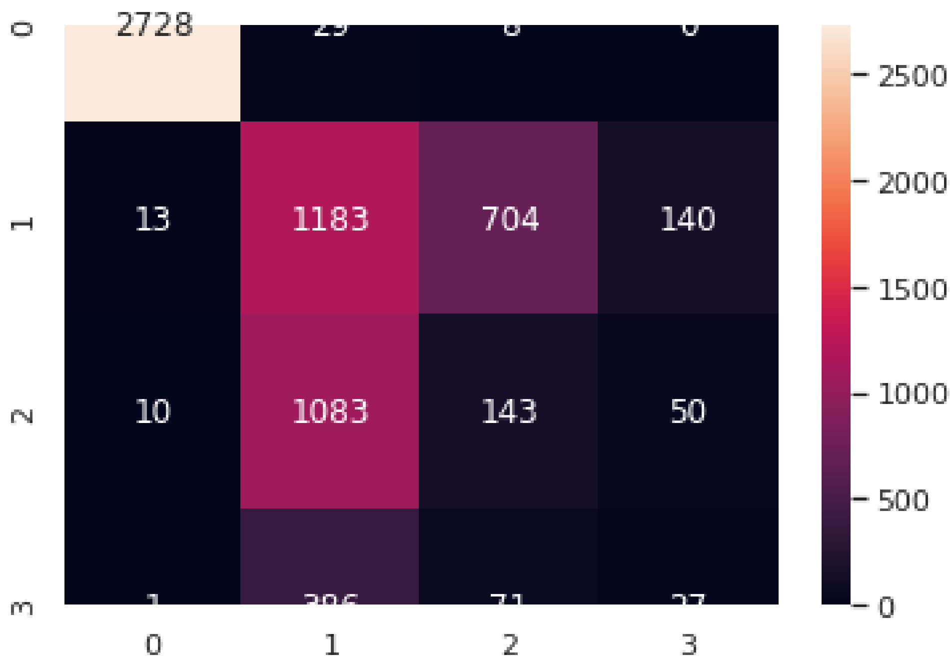
[158]:
```
print("XGB Classification Accuracy: ",metrics.accuracy_score(y_test,␣
 ↪xgb_model_predicted))
```

```
XGB Classification Accuracy:  0.6205900243309003
```

[159]:
```
print("confusion_matrix",confusion_matrix(y_test,xgb_model_predicted))
```

```
confusion_matrix [[2728   29    8    0]
 [  13 1183  704  140]
 [  10 1083  143   50]
 [   1  386   71   27]]
```

[160]:
```
uniform_data = confusion_matrix(y_test,xgb_model_predicted)
ax = sns.heatmap(uniform_data, annot=True, fmt="d")
```



[161]:
```
print("classification_report")
print(classification_report(y_test,xgb_model_predicted))
```

```
classification_report
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99      2765
         1.0       0.44      0.58      0.50      2040
         2.0       0.15      0.11      0.13      1286
         3.0       0.12      0.06      0.08       485

    accuracy                           0.62      6576
   macro avg       0.43      0.43      0.42      6576
weighted avg       0.59      0.62      0.60      6576
```

4.5 Prediction using Logistic Regression model

[162]:
```
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
```

```
/home/dheeraj/my_projects/my_project_env/lib/python3.6/site-
packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

[162]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[163]:
```
logistic_model_predicted = logistic_model.predict(X_test)
```

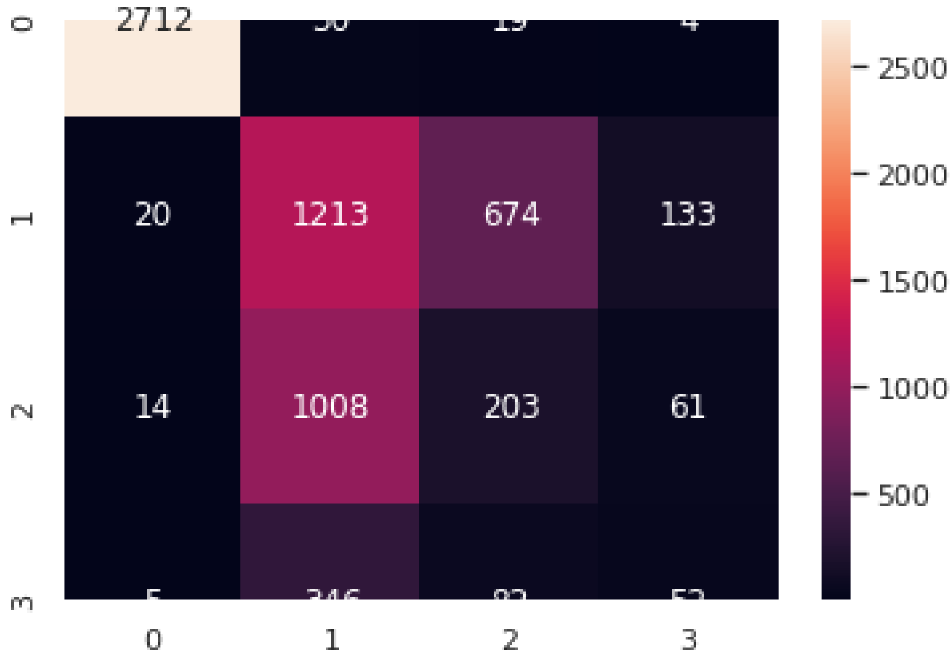
[164]:
```
print("Logistic Regression Accuracy: ",metrics.accuracy_score(y_test,␣
 ↪logistic_model_predicted))
```

```
Logistic Regression Accuracy:  0.6356447688564477
```

[165]:
```
print("confusion_matrix",confusion_matrix(y_test,logistic_model_predicted))
```

```
confusion_matrix [[2712   30   19    4]
 [  20 1213  674  133]
 [  14 1008  203   61]
 [   5  346   82   52]]
```

[166]:
```
uniform_data = confusion_matrix(y_test,logistic_model_predicted)
ax = sns.heatmap(uniform_data, annot=True, fmt="d")
```



[168]:
```
print("classification_report")
print(classification_report(y_test,logistic_model_predicted))
```

```
classification_report
              precision    recall  f1-score   support

         0.0       0.99      0.98      0.98      2765
         1.0       0.47      0.59      0.52      2040
         2.0       0.21      0.16      0.18      1286
         3.0       0.21      0.11      0.14       485

    accuracy                           0.64      6576
   macro avg       0.47      0.46      0.46      6576
weighted avg       0.62      0.64      0.62      6576
```

[ ]:
```

```

4.6 Prediction using Gradient Boosting Classifier model

[169]:
```
gradient_model = GradientBoostingClassifier()
gradient_model.fit(X_train, y_train)
```

[169]: GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)

[170]:
```
gradient_model_predicted = gradient_model.predict(X_test)
```

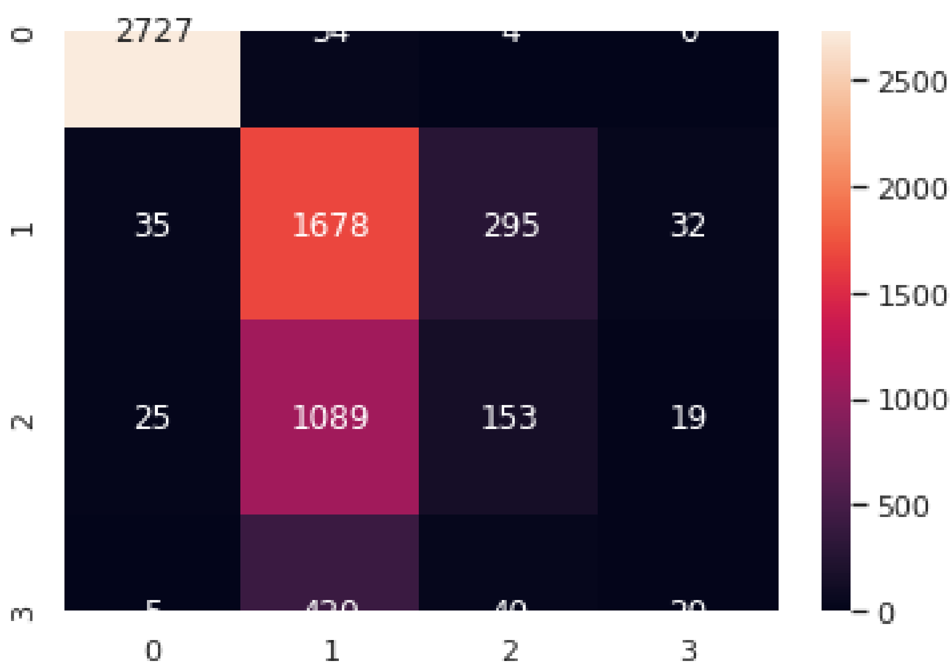
[171]:
```
print("Gradient Boosting classification Accuracy: ",metrics.
 ↪accuracy_score(y_test, gradient_model_predicted))
```

Gradient Boosting classification Accuracy: 0.6961678832116789

[172]:
```
print("confusion_matrix",confusion_matrix(y_test,gradient_model_predicted))
```

```
confusion_matrix [[2727   34    4    0]
 [  35 1678  295   32]
 [  25 1089  153   19]
 [   5  420   40   20]]
```

[173]:
```
uniform_data = confusion_matrix(y_test,gradient_model_predicted)
ax = sns.heatmap(uniform_data, annot=True, fmt="d")
```



[174]:
```python
print("classification_report")
print(classification_report(y_test,gradient_model_predicted))
```

```
classification_report
              precision    recall  f1-score   support

         0.0       0.98      0.99      0.98      2765
         1.0       0.52      0.82      0.64      2040
         2.0       0.31      0.12      0.17      1286
         3.0       0.28      0.04      0.07       485

    accuracy                           0.70      6576
   macro avg       0.52      0.49      0.47      6576
weighted avg       0.65      0.70      0.65      6576
```

[ ]:
```python

```

4.7 Prediction using Stochastic Gradient Descent model

[175]:
```python
from sklearn.linear_model import SGDClassifier
```

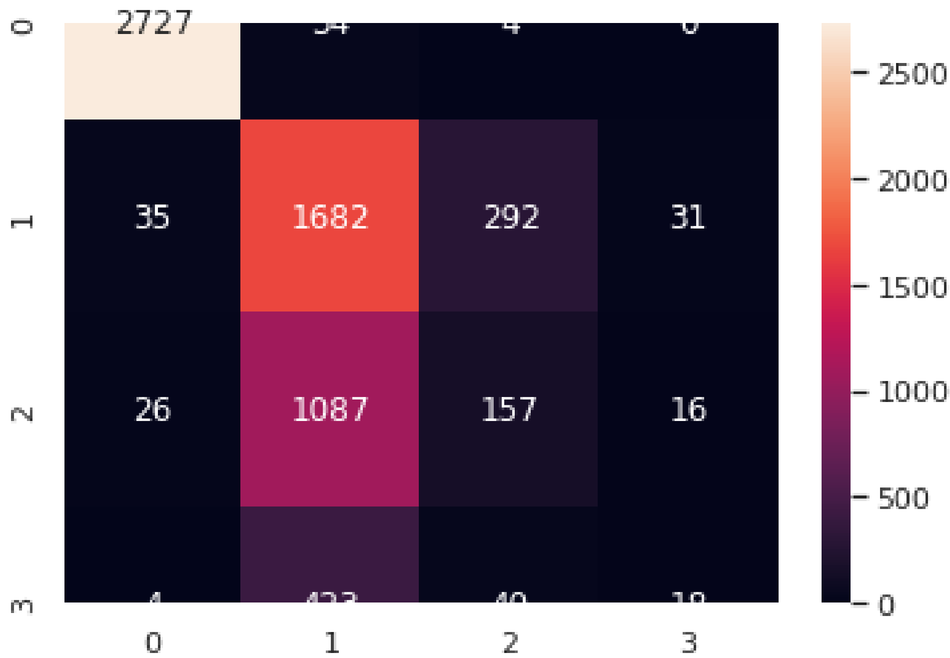
[176]:
```python
sgd_model = SGDClassifier()
sgd_model.fit(X_train, y_train)
sgd_modelpredicted = model4.predict(X_test)
print("Stochastic Gradient Descent: ",metrics.accuracy_score(y_test,␣
 ↪sgd_modelpredicted))
```

```
Stochastic Gradient Descent:  0.6970802919708029
```

[177]:
```python
print("confusion_matrix",confusion_matrix(y_test,sgd_modelpredicted))
```

```
confusion_matrix [[2727   34    4    0]
 [  35 1682  292   31]
 [  26 1087  157   16]
 [   4  423   40   18]]
```

[178]:
```python
uniform_data = confusion_matrix(y_test,sgd_modelpredicted)
ax = sns.heatmap(uniform_data, annot=True, fmt="d")
```



[179]:
```
print("classification_report")
print(classification_report(y_test,sgd_modelpredicted))
```

```
classification_report
              precision    recall  f1-score   support

         0.0       0.98      0.99      0.98      2765
         1.0       0.52      0.82      0.64      2040
         2.0       0.32      0.12      0.18      1286
         3.0       0.28      0.04      0.07       485

    accuracy                           0.70      6576
   macro avg       0.52      0.49      0.47      6576
weighted avg       0.66      0.70      0.65      6576
```

4.8 Prediction using Logistic Support Vector Machine model

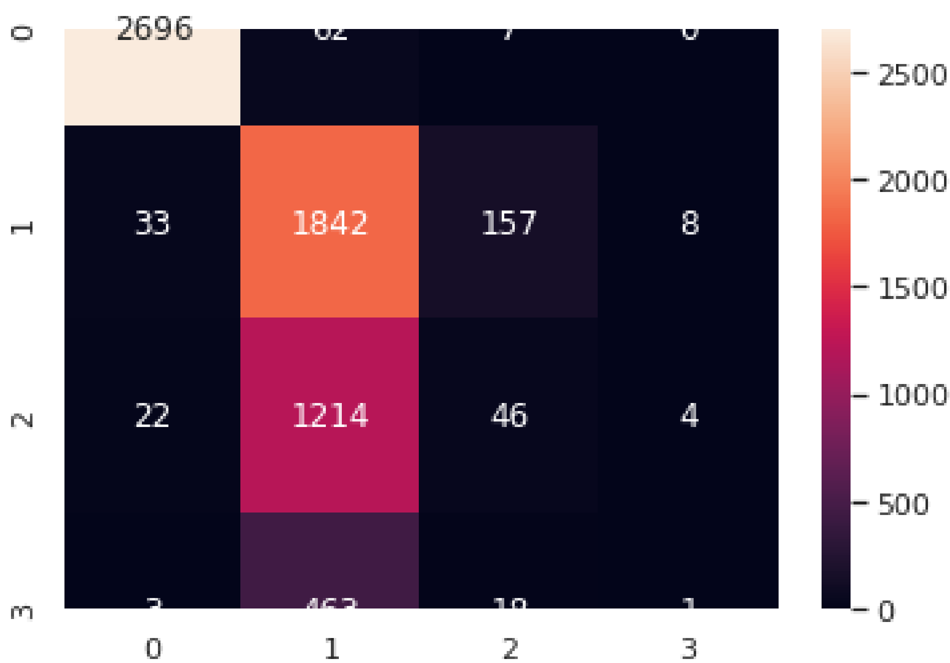
[180]:
```
from sklearn.svm import SVC
svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_modelpredicted = svm_model.predict(X_test)
print("Support Vecto Machine: ",metrics.accuracy_score(y_test,␣
 ↪svm_modelpredicted))
```

```
Support Vecto Machine:  0.6972323600973236
```

[181]:
```python
print("confusion_matrix",confusion_matrix(y_test,svm_modelpredicted))
```

```
confusion_matrix [[2696   62    7    0]
 [  33 1842  157    8]
 [  22 1214   46    4]
 [   3  463   18    1]]
```

[182]:
```python
uniform_data = confusion_matrix(y_test,svm_modelpredicted)
ax = sns.heatmap(uniform_data, annot=True, fmt="d")
```



[183]:
```python
print("classification_report")
print(classification_report(y_test,svm_modelpredicted))
```

```
classification_report
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98      2765
         1.0       0.51      0.90      0.66      2040
         2.0       0.20      0.04      0.06      1286
         3.0       0.08      0.00      0.00       485

    accuracy                           0.70      6576
   macro avg       0.44      0.48      0.42      6576
weighted avg       0.62      0.70      0.63      6576
```

5 Dataset Citation

1. Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.
2. Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127- 138