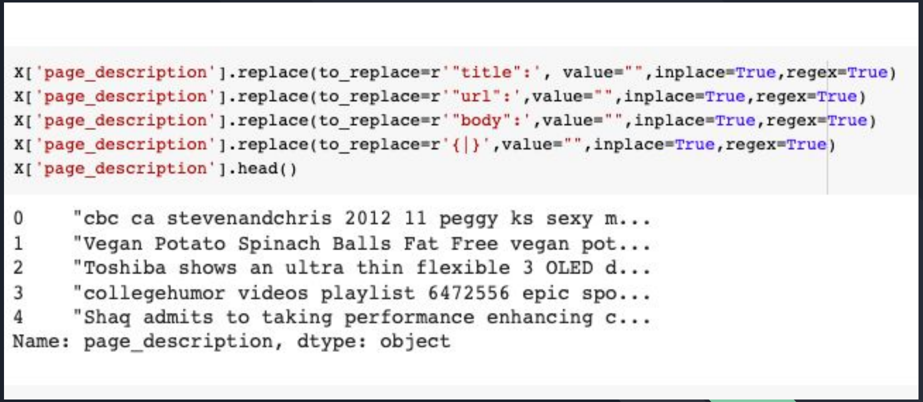
AID ESCALATING INTERNET COVERAGE

Rohan Arora: MT2022091
Sanjeev Kumar: MT2022101

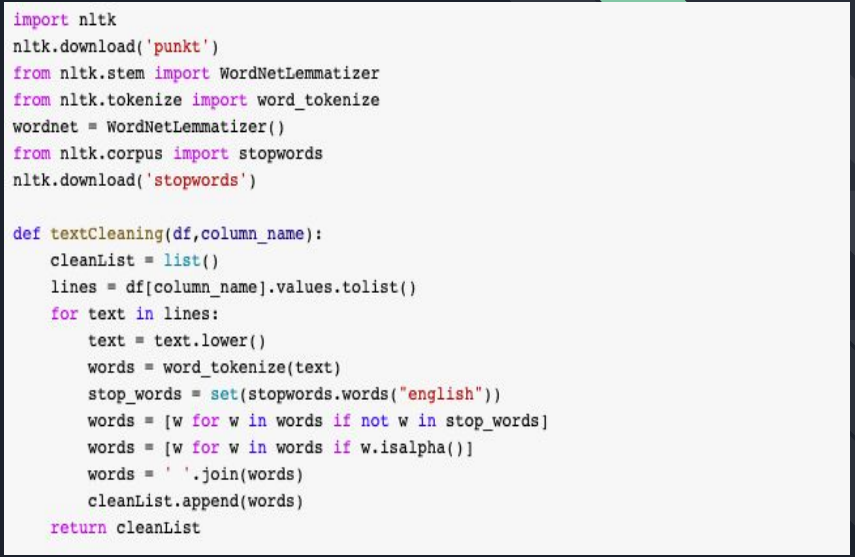
Complete walk through of the project!

NLP PREPROCESSING



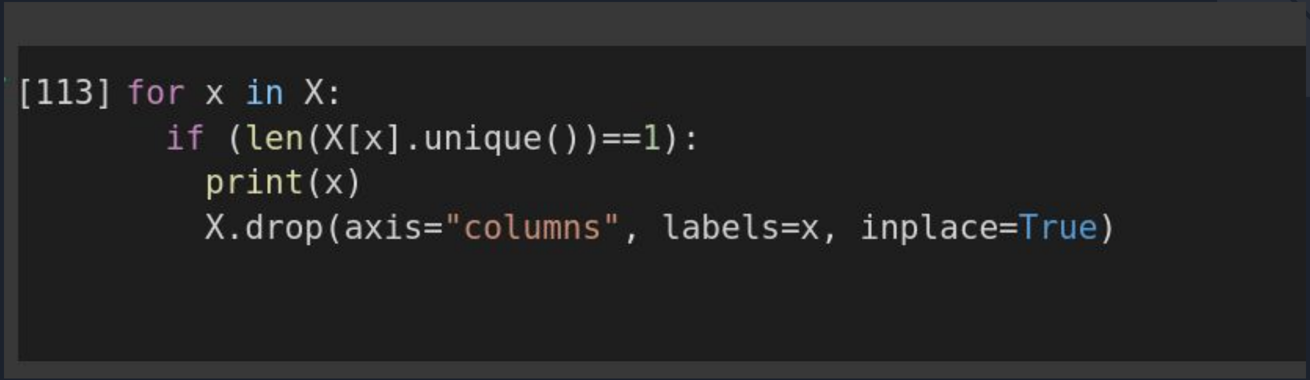
```
X['page_description'].replace(to_replace=r'"title":', value="",inplace=True,regex=True)
X['page_description'].replace(to_replace=r'"url":',value="",inplace=True,regex=True)
X['page_description'].replace(to_replace=r'"body":',value="",inplace=True,regex=True)
X['page_description'].replace(to_replace=r'{|}',value="",inplace=True,regex=True)
X['page_description'].head()
```

```
0    "cbc ca stevenandchris 2012 11 peggy ks sexy m...
1    "Vegan Potato Spinach Balls Fat Free vegan pot...
2    "Toshiba shows an ultra thin flexible 3 OLED d...
3    "collegehumor videos playlist 6472556 epic spo...
4    "Shaq admits to taking performance enhancing c...
Name: page_description, dtype: object
```



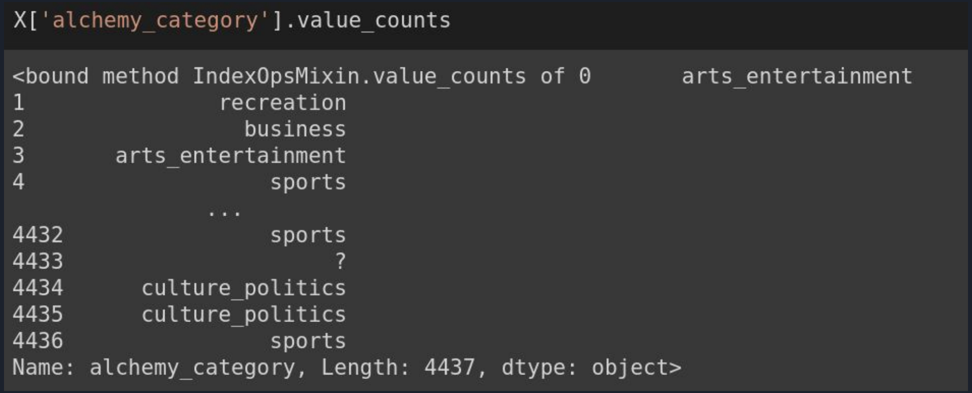
```
import nltk
nltk.download('punkt')
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
wordnet = WordNetLemmatizer()
from nltk.corpus import stopwords
nltk.download('stopwords')

def textCleaning(df,column_name):
    cleanList = list()
    lines = df[column_name].values.tolist()
    for text in lines:
        text = text.lower()
        words = word_tokenize(text)
        stop_words = set(stopwords.words("english"))
        words = [w for w in words if not w in stop_words]
        words = [w for w in words if w.isalpha()]
        words = ' '.join(words)
        cleanList.append(words)
    return cleanList
```

Dropping columns with non unique values



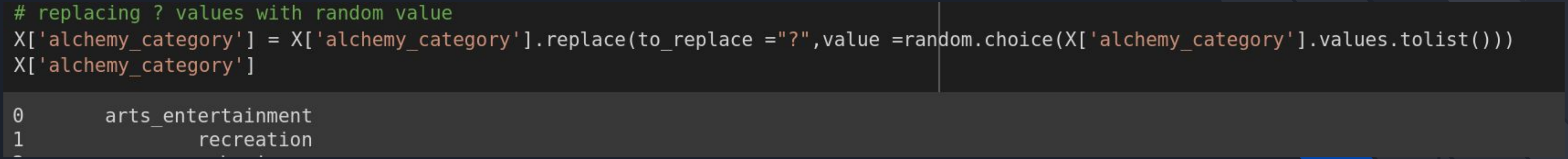
```
[113] for x in X:
        if (len(X[x].unique())==1):
          print(x)
          X.drop(axis="columns", labels=x, inplace=True)
```

Checking null values:



```
X['alchemy_category'].value_counts
```

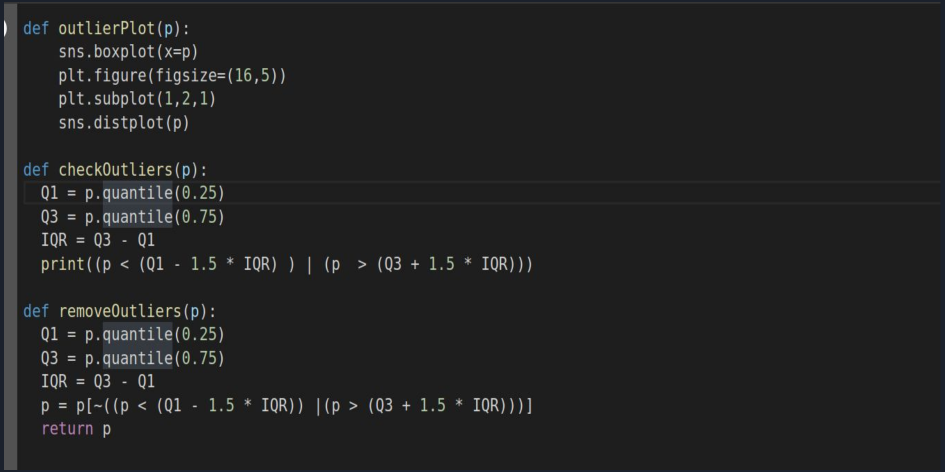
```
<bound method IndexOpsMixin.value_counts of 0       arts_entertainment
1                recreation
2                  business
3        arts_entertainment
4                    sports
               ...
4432                 sports
4433                      ?
4434       culture_politics
4435       culture_politics
4436                 sports
Name: alchemy_category, Length: 4437, dtype: object>
```



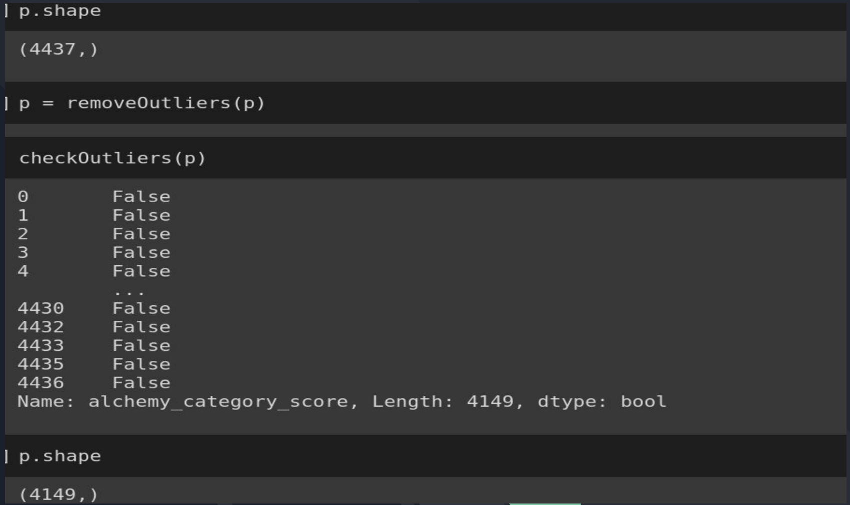
```
# replacing ? values with random value
X['alchemy_category'] = X['alchemy_category'].replace(to_replace ="?",value =random.choice(X['alchemy_category'].values.tolist()))
X['alchemy_category']
```

```
0       arts_entertainment
1               recreation
```

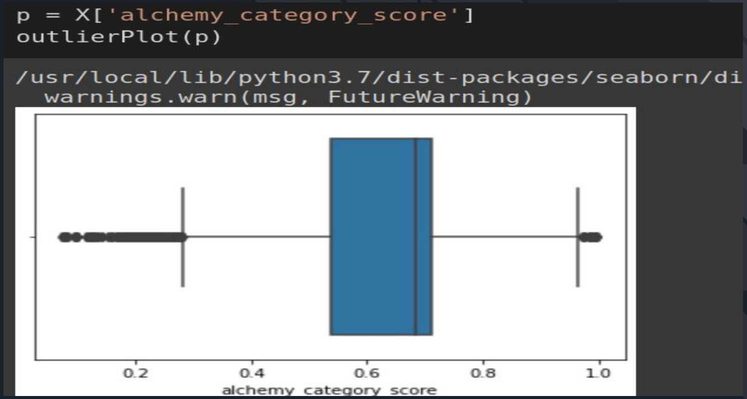
Removing Outliers:



```
def outlierPlot(p):
    sns.boxplot(x=p)
    plt.figure(figsize=(16,5))
    plt.subplot(1,2,1)
    sns.distplot(p)

def checkOutliers(p):
  Q1 = p.quantile(0.25)
  Q3 = p.quantile(0.75)
  IQR = Q3 - Q1
  print((p < (Q1 - 1.5 * IQR) ) | (p  > (Q3 + 1.5 * IQR)))

def removeOutliers(p):
  Q1 = p.quantile(0.25)
  Q3 = p.quantile(0.75)
  IQR = Q3 - Q1
  p = p[~((p < (Q1 - 1.5 * IQR)) |(p > (Q3 + 1.5 * IQR)))]
  return p
```

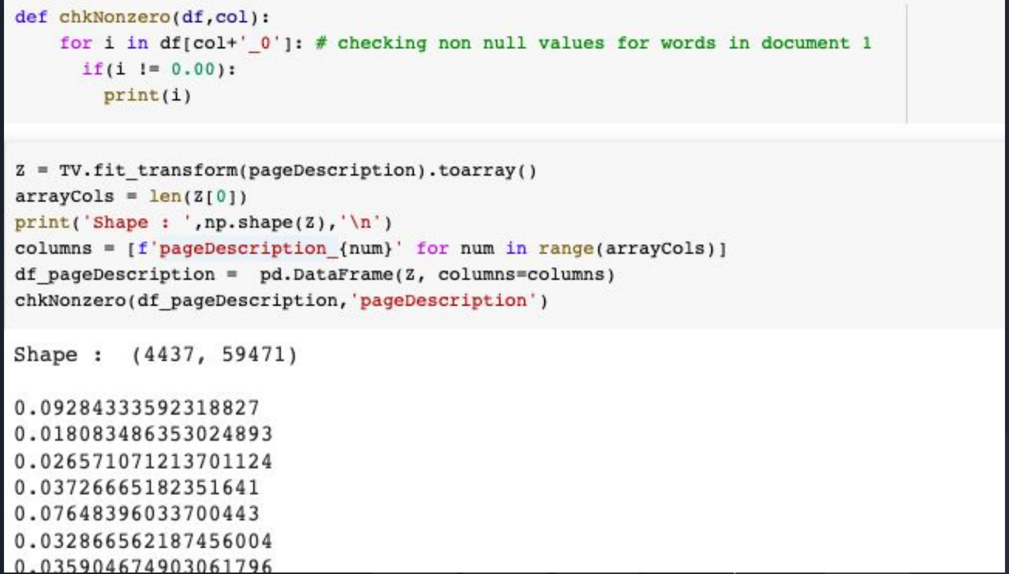


```
] p.shape

(4437,)

] p = removeOutliers(p)

checkOutliers(p)

0       False
1       False
2       False
3       False
4       False
        ...
4430    False
4432    False
4433    False
4435    False
4436    False
Name: alchemy_category_score, Length: 4149, dtype: bool

] p.shape

(4149,)
```



```
p = X['alchemy_category_score']
outlierPlot(p)

/usr/local/lib/python3.7/dist-packages/seaborn/di
  warnings.warn(msg, FutureWarning)
```

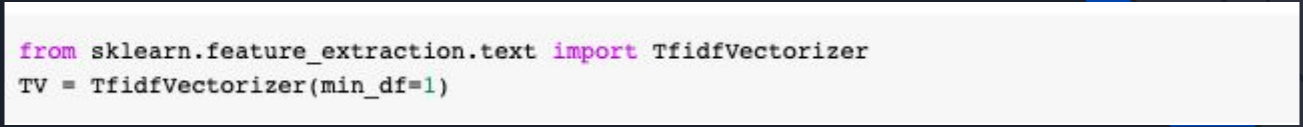
TFIDF VECTORIZATION



```
def chkNonzero(df,col):
    for i in df[col+'_0']: # checking non null values for words in document 1
      if(i != 0.00):
        print(i)
```

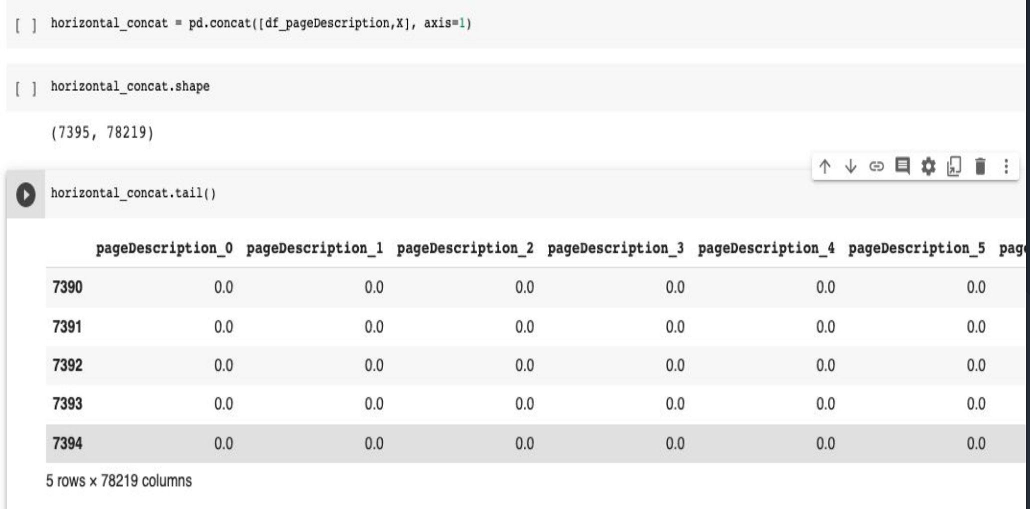
```
Z = TV.fit_transform(pageDescription).toarray()
arrayCols = len(Z[0])
print('Shape : ',np.shape(Z),'\n')
columns = [f'pageDescription_{num}' for num in range(arrayCols)]
df_pageDescription =  pd.DataFrame(Z, columns=columns)
chkNonzero(df_pageDescription,'pageDescription')
```

```
Shape :  (4437, 59471)

0.09284333592318827
0.018083486353024893
0.026571071213701124
0.03726665182351641
0.07648396033700443
0.032866562187456004
0.035904674903061796
```



```
from sklearn.feature_extraction.text import TfidfVectorizer
TV = TfidfVectorizer(min_df=1)
```

A little overhead by merging the dataset obtained from tf-idf with the traditional database and we get the new database of whole e new amount of columns



```
[ ] horizontal_concat = pd.concat([df_pageDescription,X], axis=1)
```

```
[ ] horizontal_concat.shape
```

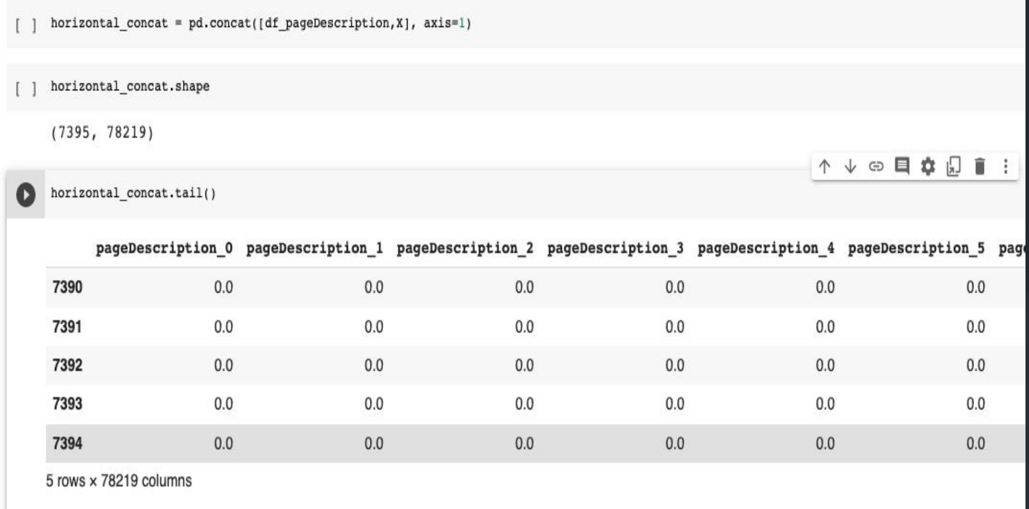
```
(7395, 78219)
```

```
horizontal_concat.tail()
```

	pageDescription_0	pageDescription_1	pageDescription_2	pageDescription_3	pageDescription_4	pageDescription_5	pag
7390	0.0	0.0	0.0	0.0	0.0	0.0	
7391	0.0	0.0	0.0	0.0	0.0	0.0	
7392	0.0	0.0	0.0	0.0	0.0	0.0	
7393	0.0	0.0	0.0	0.0	0.0	0.0	
7394	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 78219 columns

A little overhead by merging the dataset obtained from tf-idf with the traditional database and we get the new database of whole e new amount of columns



```
[ ] horizontal_concat = pd.concat([df_pageDescription,X], axis=1)
```

```
[ ] horizontal_concat.shape
```

```
(7395, 78219)
```

```
horizontal_concat.tail()
```

	pageDescription_0	pageDescription_1	pageDescription_2	pageDescription_3	pageDescription_4	pageDescription_5	pag
7390	0.0	0.0	0.0	0.0	0.0	0.0	
7391	0.0	0.0	0.0	0.0	0.0	0.0	
7392	0.0	0.0	0.0	0.0	0.0	0.0	
7393	0.0	0.0	0.0	0.0	0.0	0.0	
7394	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows x 78219 columns

Random forest results:
87.26%

Logistic Regression
87.428%

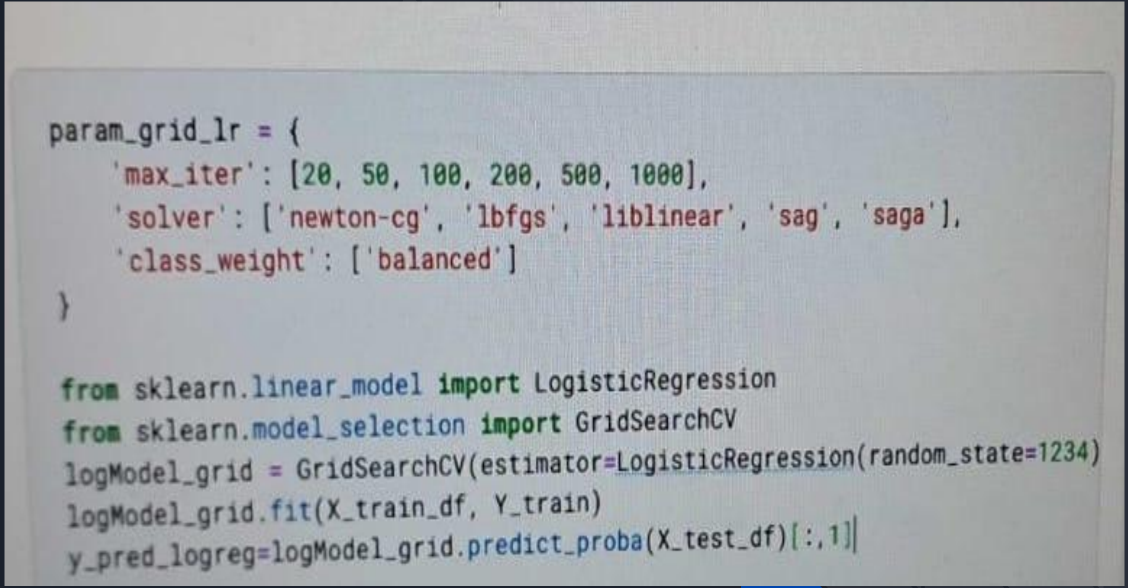
XG-BOOST
87.33%

Linear SVM
86.742%

Logistic Regression Approach

- Max_iteration parameter that best suited was 1500
- Trained without normalization and with normalization and result without normalization was high as compared to with normalization
- Concatenated tf0idf data approach suited in case of logistic regression
- Adding hyper parameter of equal weight in logistic regression could not help much

Tried adding advance level hyper parameter tuning but could train model as it was crashing on various platforms



```
param_grid_lr = {
    'max_iter': [20, 50, 100, 200, 500, 1000],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'class_weight': ['balanced']
}

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
logModel_grid = GridSearchCV(estimator=LogisticRegression(random_state=1234)
logModel_grid.fit(X_train_df, Y_train)
y_pred_logreg=logModel_grid.predict_proba(X_test_df)[:,1]
```