National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Apurva Kumari
Student ID: X21118175

School of Computing
National College of Ireland

Supervisor: Mr Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Apurva Kumari
Student ID:	X21118175
Programme:	Data Analytics
Year:	2022
Module:	Research Project
Supervisor:	Mr Bharat Agarwal
Submission Due Date:	15/12/2022
Project Title:	Exploration of the Most Preferred Social Media for the Fashion Business Practices
Word Count:	1062
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Apurva Kumari
Date:	15th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Apurva Kumari
X21118175

1 Introduction

For the purposes of predicting business growth for fashion businesses and customer satisfaction, this paper shows how to categorize customer reviews so that they can be used again. Here are the procedures and specifications needed to recreate the machine learning models.

2 System Configuration

The necessary hardware and software settings for the research are detailed below, along with illustrations depicting them.

2.1 Hardware Configuration

Regarding the hardware configuration, an HP laptop with a 1.20 GHz Intel Core i3-1005G1 processor, 8 gigabytes of random access memory, and a 64-bit operating system was utilized (see Figure 1).

Item	Value
OS Name	Microsoft Windows 10 Home Single Language
Version	10.0.19045 Build 19045
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	LAPTOP-6FA4F31U
System Manufacturer	HP
System Model	HP Laptop 14s-dr1xxx
System Type	x64-based PC
System SKU	227Q3PA#ACJ
Processor	Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz, 1190 ...
BIOS Version/Date	AMI F.25, 09-03-2022
SMBIOS Version	3.2
Embedded Controll...	56.33
BIOS Mode	UEFI
BaseBoard Manufact...	HP
BaseBoard Product	86C8
BaseBoard Version	56.33
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\windows
System Directory	C:\windows\system32

Figure 1: Hardware Configuration

2.2 Software Configuration

Several different programs, such as Jupyter notebook, Microsoft Excel, Power Bi, and Twitter API Setup, have been used to set up software. Figures 2 and 3 show, for the Twitter dataset, how an account is made and how an API key is gotten. The version of Jupyter Notebook that works with Anaconda Navigator is shown in Figure 2.
1. Jupyter Notebook and Anaconda Navigator : Various python libraries/packages were installed beforehand either in anaconda environment or Jupyter notebook for sucessfull implementation of whole code.
The majorpackages/libraries are: Tweepy,Pandas,TextBlob,NLTK, Numpy,WordCloud, Seaborn, Transformers,Tensorflow,Sklearn,Tensorflow-gpu, Hyperopt,Emoji,Imblearn.
2. Microsoft Excel : All datasets have been saved here.
3. Power BI : With the help of this software, exploratory data analysis and data visualization have been done.
4. Twitter API account setup and API key generation : Few steps needs to be completed before accessing authenticated Twitter APIs defined below::

• Apply and receive approval for Twitter developer account[1] shown in Figure 2.

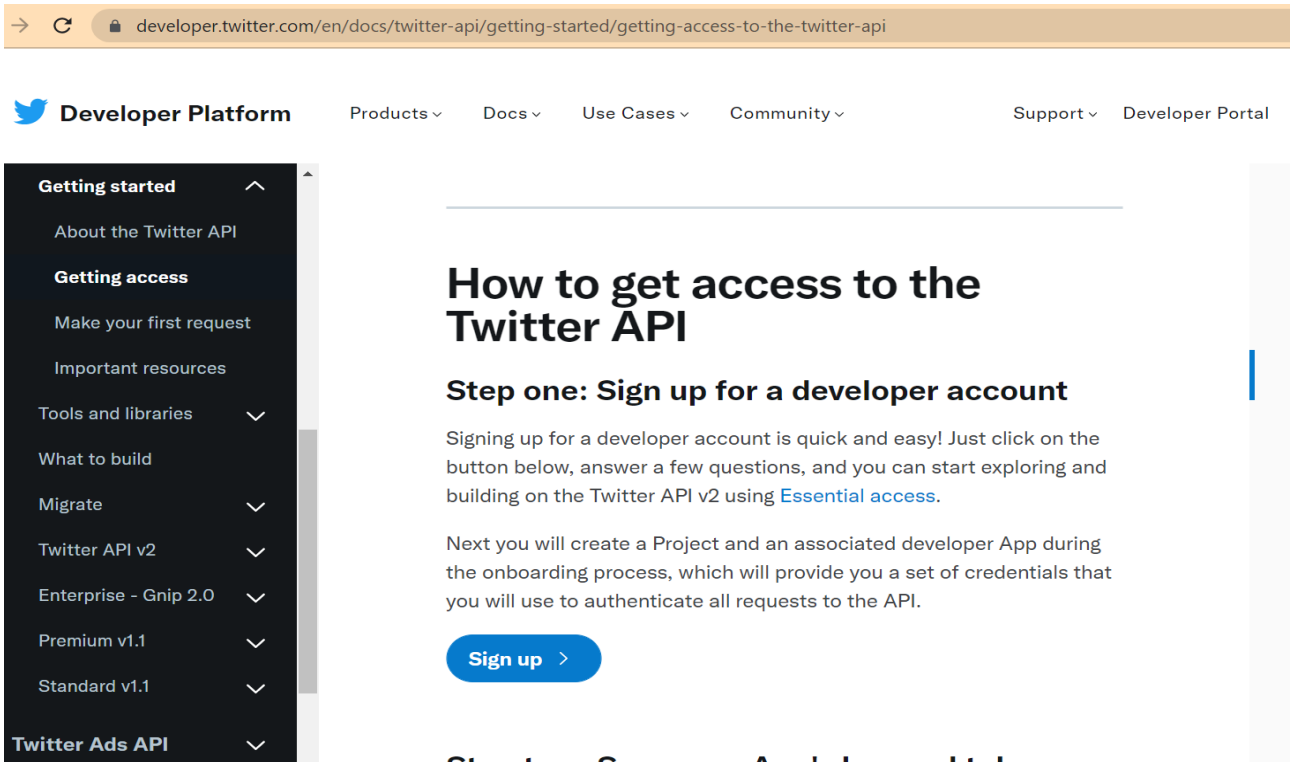

Figure 2: Developer account signup

• Develop your own Twitter application account.
• Make sure your project has its own unique API keys and access tokens by generating them shown in Figure 3.

[1] https://developer.twitter.com/en

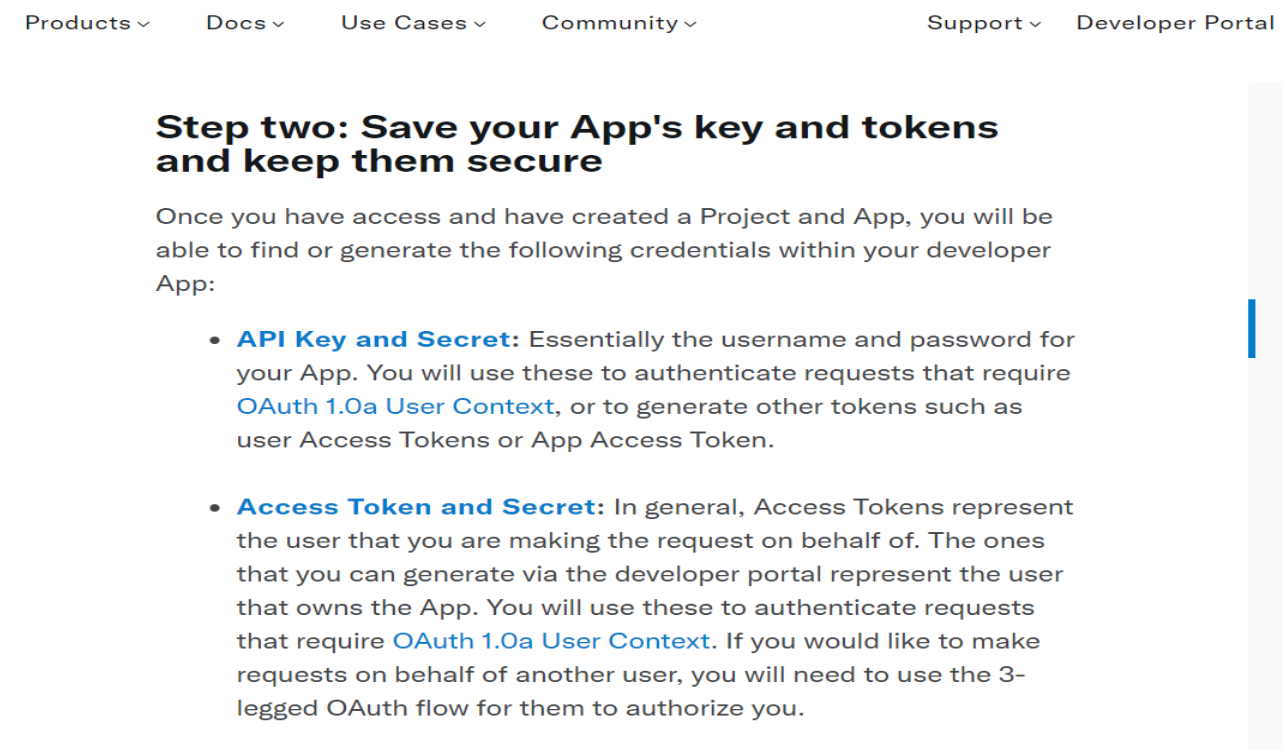

Figure 3: API keys and token generation

- Generate bearer tokens for your app.

- Apply for access to the specified API and receive it as shown in Figure 4 and 5.

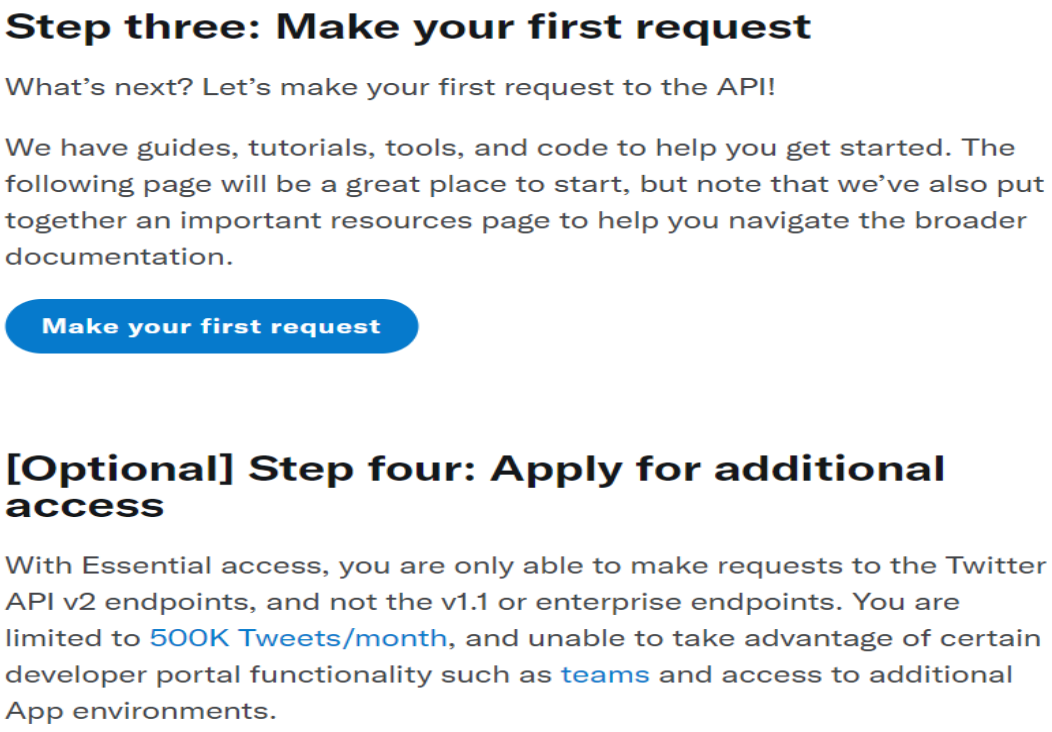

Figure 4: Final request for Twitter API

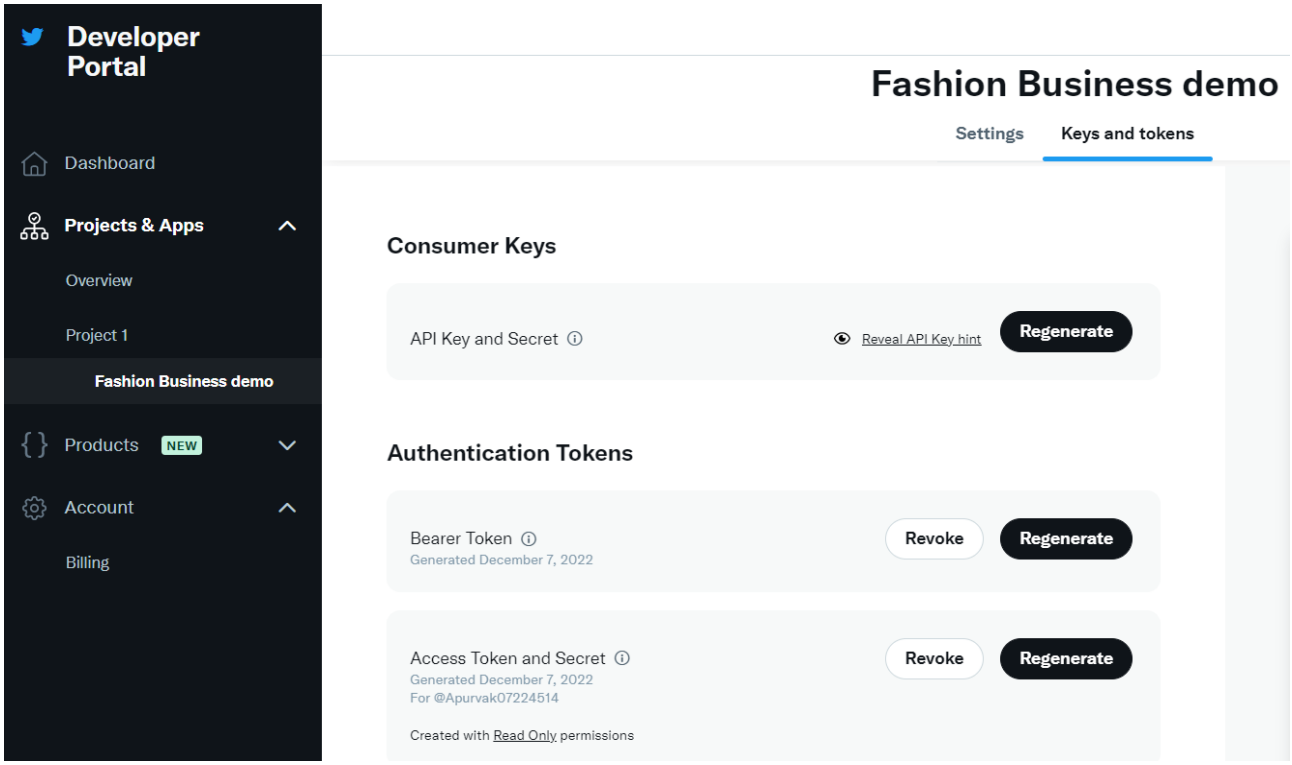

Figure 5: Final created Twitter APP and developer portal

3 Implementation

3.1 Source of data

The dataset was fetched using Twitter API and creating twitter developer account. The link used for the same was: https://developer.twitter.com/en

3.2 Data Cleaning/Preprocessing/Transformation

1. Data was fetched from Twitter using Twitter API with the help of API keys and Tokens shown in below Figures 6 and 7. In figure 6, python library Tweepy is imported. to extract data from Twitter. API access keys and tokens were used to fetch texts from Twitter using different hashtags related to fashion business. Data fetched using hashtags is then saved in csv file.

```
In [2]: import pandas as pd
        import tweepy

        # function to display data of each tweet
        def printtweetdata(n, ith_tweet):
                print()
                print(f"Tweet {n}:")
                print(f"Username:{ith_tweet[0]}")
                print(f"Description:{ith_tweet[1]}")
                print(f"Location:{ith_tweet[2]}")
                print(f"Following Count:{ith_tweet[3]}")
                print(f"Follower Count:{ith_tweet[4]}")
                print(f"Total Tweets:{ith_tweet[5]}")
                print(f"Retweet Count:{ith_tweet[6]}")
                print(f"Tweet Text:{ith_tweet[7]}")
                print(f"Hashtags Used:{ith_tweet[8]}")


        # function to perform data extraction
        def scrape(words, date_since, numtweet):

                # Creating DataFrame using pandas
                db = pd.DataFrame(columns=['username',
                                           'description',
                                           'location',
                                           'following',
                                           'followers',
                                           'totaltweets',
                                           'retweetcount',
                                           'text',
                                           'hashtags'])

                # We are using .Cursor() to search
                # through twitter for the required tweets.
                # The number of tweets can be
                # restricted using .items(number of tweets)
                tweets = tweepy.Cursor(api.search_tweets,
                                       words, lang="en",
                                       since_id="2020-01-01",
                                       tweet_mode='extended').items(numtweet)


                # .Cursor() returns an iterable object. Each item in
                # the iterator has various attributes
                # that you can access to
                # get information about each tweet
                list_tweets = [tweet for tweet in tweets]

                # Counter to maintain Tweet Count
                i = 1

                # we will iterate over each tweet in the
                # list for extracting information about each tweet
                for tweet in list_tweets:
                        username = tweet.user.screen_name
                        description = tweet.user.description
                        location = tweet.user.location
                        following = tweet.user.friends_count
                        followers = tweet.user.followers_count
                        totaltweets = tweet.user.statuses_count
```

Figure 6: Twitter data extracted using Tweepy

```
if __name__ == '__main__':

        # Enter your own credentials obtained
        # from your developer account
        consumer_key = "MDRP2mJVfbIF4TYfjAiEdX7xC"
        consumer_secret = "qvDpk8w9v5lGyN4UsfefXYlKfcGrTSb3Erdho3FNnQNt9kKxHs"
        access_key = "1600501539046625286-DBmtPFq4yog50hdG5pi4LTuIWSnk20"
        access_secret = "oJ6VRoA2zl0tABhWBZz8khJkIkVBPf99Z66KxywdkyU5o"


        auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_key, access_secret)
        api = tweepy.API(auth)

        # Enter Hashtag and initial date
        print("Enter Twitter HashTag to search for")
        words = input()
        print("Enter Date since The Tweets are required in yyyy-mm--dd")
        date_since = "2020-01-01"

        # number of tweets you want to extract in one run
        numtweet = 1000
        scrape(words, date_since, numtweet)
        print('Scraping has completed!')
```

Figure 7: Data extracted using Access keys and Tokens

2. Using the previous code, which incorporated the four most popular hashtags, the data was saved in the.csv file. Using the pandas package, the code seen in Figures 8 below turned all of the datasets' data into a structured format using the DataFrame object in Python. Also, all 4 dataframes are selected using head() function for first 5 records.

```
In [1]: import pandas as pd
        df1=pd.read_csv('#fashion.csv')
        df2=pd.read_csv('#fashionbrand.csv')
        df3=pd.read_csv('#onlineshopping.csv')
        df4=pd.read_csv('#clothingbrand.csv')

In [2]: df1.head()
```

Out[2]:

	Unnamed: 0	username	description	location	following	followers	totaltweets	retweetcount	text	hashtags
0	0	TheEventFundis	With us "you" are the brand. Clothing design h...	Pretoria	87	61	133	0	Welcome to the Prestige Sports Awards 2022 dre...	['bosslady', 'GirlBuyZa', 'GirlTalkZA', 'shopp...
1	1	buy1_best	Always #young #ladies are here. Join us now. B...	New York, NY	8023	9348	635146	0	👉 $206.26 👈 \nIrregular Choice Amore Womens Syn...	['Irregular', 'Choice', 'Amore', 'Womens', 'Sy...
2	2	snkrscopstore	🥤 Hottest Sneaker Drop 50% OFF SALE!!\n🌎 World...	NaN	20	51	4793	0	Free Sneakers Giveaway!!! Nike Air Jordan 1 ‼...	['nike', 'sneakerhead', 'offwhite', 'sneakers'...
3	3	DESIblitz	News | Gossip | Gupshup Discover What's Trendi...	Birmingham, England	479	6707	92120	0	10 Top Casual looks of Bollywood Actors\n\nVer...	['fashion', 'style', 'bollywood']
4	4	AmazingArchi	A leading platform for #architecture projects ...	Mexico	200	9088	12862	0	Ha'Pelech, Tel Aviv, Israel by Erez Shani Arch...	['architecture', 'house', 'fashion', 'decor', ...

Figure 8: Data frames created

3. Various libraries like textblob[2], scikit-learn[3] and NLTk[4] are imported for data-preprocessing of texts for sentiment analysis.
4. All the tweets extracted are listed into three categories based on tones and sentiments of texts namely, neutral,negative and positive. Total percent of each sentiment is fetched in below Figure 9.

```
In [15]: tw_list_negative = tw_list[tw_list["sentiment"]=="negative"]
         tw_list_positive = tw_list[tw_list["sentiment"]=="positive"]
         tw_list_neutral = tw_list[tw_list["sentiment"]=="neutral"]

In [16]: def count_values_in_column(data,feature):
             total=data.loc[:,feature].value_counts(dropna=False)
             percentage=round(data.loc[:,feature].value_counts(dropna=False,normalize=True)*100,2)
             return pd.concat([total,percentage],axis=1,keys=['Total','Percentage'])

In [17]: #Count_values for sentiment
         count_values_in_column(tw_list,"sentiment")
```

Out[17]:

	Total	Percentage
positive	1928	61.81
neutral	1026	32.90
negative	165	5.29

Figure 9: Texts classified into sentiments

5. In Figure 10, data cleaning and transformation is performed on classified texts where all punctuations are removed, tokenization is applied on texts , stopwords are removed from texts and Stemming is performed. Next, In figure 11 countvectorizer is performed to convert all texts into mathematical format in NLP to train model.A matrix of words(3319,6553) is formed in the end to apply it in model.

[2]TextBlob analyzes emotions using the Lexicon. It includes polarity scores, rules, and a word-weight dictionary.

[3]Python module used in sentiment analysis to detect the underlying emotional tone of phrases using Python-written computational algorithms.

[4]NLTK is a set of Python tools and programs for symbolic and statistical natural language processing in English.

In [28]:
```
#Removing Punctuation
def remove_punct(text):
    text  = "".join([char for char in text if char not in string.punctuation])
    text = re.sub('[0-9]+', '', text)
    return text

tw_list['punct'] = tw_list['text'].apply(lambda x: remove_punct(x))

#Appliyng tokenization
def tokenization(text):
    text = re.split('\W+', text)
    return text

tw_list['tokenized'] = tw_list['punct'].apply(lambda x: tokenization(x.lower()))
#Removing stopwords
stopword = nltk.corpus.stopwords.words('english')
def remove_stopwords(text):
    text = [word for word in text if word not in stopword]
    return text

tw_list['nonstop'] = tw_list['tokenized'].apply(lambda x: remove_stopwords(x))
#Appliyng Stemmer
ps = nltk.PorterStemmer()

def stemming(text):
    text = [ps.stem(word) for word in text]
    return text

tw_list['stemmed'] = tw_list['nonstop'].apply(lambda x: stemming(x))
#Cleaning Text
def clean_text(text):
    text_lc = "".join([word.lower() for word in text if word not in string.punctuation]) # remove puntuation
    text_rc = re.sub('[0-9]+', '', text_lc)
    tokens = re.split('\W+', text_rc)    # tokenization
    text = [ps.stem(word) for word in tokens if word not in stopword]  # remove stopwords and stemming
    return text
tw_list.head()
```

Figure 10: Twitter data Pre-processing

In [29]:
```
#Appliyng Countvectorizer
countVectorizer = CountVectorizer(analyzer=clean_text)
countVector = countVectorizer.fit_transform(tw_list['text'])
print('{} Number of reviews has {} words'.format(countVector.shape[0], countVector.shape[1]))
#print(countVectorizer.get_feature_names())
count_vect_df = pd.DataFrame(countVector.toarray(), columns=countVectorizer.get_feature_names_out())
count_vect_df.head()
```

```
3119 Number of reviews has 6553 words
```

Out[29]:

	aajeevika	aandh	aaronhasmoney	abandon	abbey	abercrombiefitch	abercrombiekid	abound	absolut	...	zebr	zebra	zenpet	zerodownpay	zeromaria
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

5 rows × 6553 columns

Figure 11: Countvectorizer applied

6. The clean data is then exported to csv file saved as **all_tweets_fashion.csv**.
7. Next, data is transformed further separately for BERT model using cleaned texts.

Bert is a pre-trained model where Bert weights are modeified as per Twitter dataset. Token(Most occuring texts) lengths from tweet sentences are fetched and added in the datasets which is further used in Bert model. Emojis are removed from clean texts by importing emoji library. Various libraries were imported to successfully implement BERT classifier like transformers, tensorflow, tensorflow-gpu, rich=12.0.1 ,sklearn etc.
Below Figure 12 shows final architecture of BERT classifier where number of total, trainable and non-trainable parameters are defined

```
In [59]: model = create_model(bert_model, MAX_LEN)
         model.summary()

Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 128)]        0           []

 input_2 (InputLayer)           [(None, 128)]        0           []

 tf_bert_model (TFBertModel)    TFBaseModelOutputWi  109482240   ['input_1[0][0]',
                                thPoolingAndCrossAt               'input_2[0][0]']
                                tentions(last_hidde
                                n_state=(None, 128,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 dense (Dense)                  (None, 3)            2307        ['tf_bert_model[0][1]']

==================================================================================================
Total params: 109,484,547
Trainable params: 109,484,547
Non-trainable params: 0
```

Figure 12: BERT architecture

8. Lastly, imblearn library is used to balance the classified data by implementing oversampling technique and Labelencoder() is performed to label classified texts where Neutral is encoded as 0, Positive as 1 and Negative as 2 respectively.
9. In below Figure 13, the first model implemented is Naive Bayes Classifier where Countervectorized matrix (3119,7786) is used in training models as X_features and confusion matrix is fetched.

```
In [55]: from sklearn.metrics import classification_report, confusion_matrix
         y_pred_test = NaiveBclassifier.predict(X_test)
         # Confusion matrix
         cm = confusion_matrix(y_test, y_pred_test)
         sns.heatmap(cm, annot= True)
```

Figure 13: Naive Bayes Model

10. In below Figure 14, the second model to be implemented is XGBoost classifier where hyperparameter tuning is also performed to fetch best parameters to obtain better accuracy.

In [63]:
```
###xgboost classifier
import xgboost as xgb
space={'max_depth': hp.quniform("max_depth", 3, 18, 1),
        'gamma': hp.uniform ('gamma', 1,9),
        'reg_alpha' : hp.quniform('reg_alpha', 40,180,1),
        'reg_lambda' : hp.uniform('reg_lambda', 0,1),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0.5,1),
        'min_child_weight' : hp.quniform('min_child_weight', 0, 10, 1),
        'n_estimators': 180,
        'seed': 0
    }
def objective(space):
    clf=xgb.XGBClassifier(
                    n_estimators =space['n_estimators'], max_depth = int(space['max_depth']), gamma = space['gamma'],
                    reg_alpha = int(space['reg_alpha']),min_child_weight=int(space['min_child_weight']),
                    colsample_bytree=int(space['colsample_bytree']))

    evaluation = [( X_train, y_train), ( X_test, y_test)]

    clf.fit(X_train, y_train,
            eval_set=evaluation, eval_metric="auc",
            early_stopping_rounds=10,verbose=False)


    pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, pred>0.5)
    print ("SCORE:", accuracy)
    return {'loss': -accuracy, 'status': STATUS_OK }
```

In [64]:
```
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
trials = Trials()

best_hyperparams = fmin(fn = objective,
                        space = space,
                        algo = tpe.suggest,
                        max_evals = 100,
                        trials = trials)
```

Figure 14: XGBoost Classifier

11. In below Figure 15, third mode , SVM (Support Vector Machine) is performed where GridSearchCV is used for hyperparameter tuning where best possible tunable paprameters are estimated and accuracy is optimized.

In [70]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid']}
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
grid.fit(X_train,y_train)
print(grid.best_estimator_)
```

Figure 15: SVM Classifier

12. In below Figure 16, BERT model classifier is performed where trained data is masked and tokenized length is used for training the model. In order to achieve optimized accuracy four epochs are run whcih take some time to run.

```
In [102]: history_bert = model.fit([train_input_ids,train_attention_masks], y_train, validation_data=([val_input_ids,val_attention_masks],

Epoch 1/4
163/163 [==============================] - 6130s 37s/step - loss: 0.5084 - categorical_accuracy: 0.7881 - val_loss: 0.1914 - va
l_categorical_accuracy: 0.9292
Epoch 2/4
163/163 [==============================] - 6121s 38s/step - loss: 0.1334 - categorical_accuracy: 0.9535 - val_loss: 0.0838 - va
l_categorical_accuracy: 0.9706
Epoch 3/4
163/163 [==============================] - 7593s 47s/step - loss: 0.0604 - categorical_accuracy: 0.9793 - val_loss: 0.0410 - va
l_categorical_accuracy: 0.9879
Epoch 4/4
163/163 [==============================] - 6892s 42s/step - loss: 0.0222 - categorical_accuracy: 0.9933 - val_loss: 0.0452 - va
l_categorical_accuracy: 0.9862

In [103]: result_bert = model.predict([val_input_ids,val_attention_masks])

19/19 [==============================] - 156s 8s/step

In [104]: y_pred_bert =  np.zeros_like(result_bert)
y_pred_bert[np.arange(len(y_pred_bert)), result_bert.argmax(1)] = 1
```

Figure 16: BERT model Classifier

3.3 Evaluation

1. Figure 17 depicts a piece of code that has been used to demonstrate numerous classification performance measurements. According to these data, the BERT model classifier appears to be the most accurate.

```
In [105]: print(classification_report(y_valid.argmax(1), y_pred_bert.argmax(1)))

              precision    recall  f1-score   support

           0       0.99      1.00      1.00       193
           1       0.98      0.98      0.98       193
           2       0.98      0.97      0.98       193

    accuracy                           0.99       579
   macro avg       0.99      0.99      0.99       579
weighted avg       0.99      0.99      0.99       579
```

Figure 17: BERT model Classification results

2. Below Figure 18, shows confusion matrix for BERT model classifier where 0,1,2 depicts Neutral, Positive ,Negative respectively.

```
In [106]: cm_xg = confusion_matrix(y_valid.argmax(1), y_pred_bert.argmax(1))
          sns.heatmap(cm_xg, annot= True)
Out[106]: <AxesSubplot: >
```

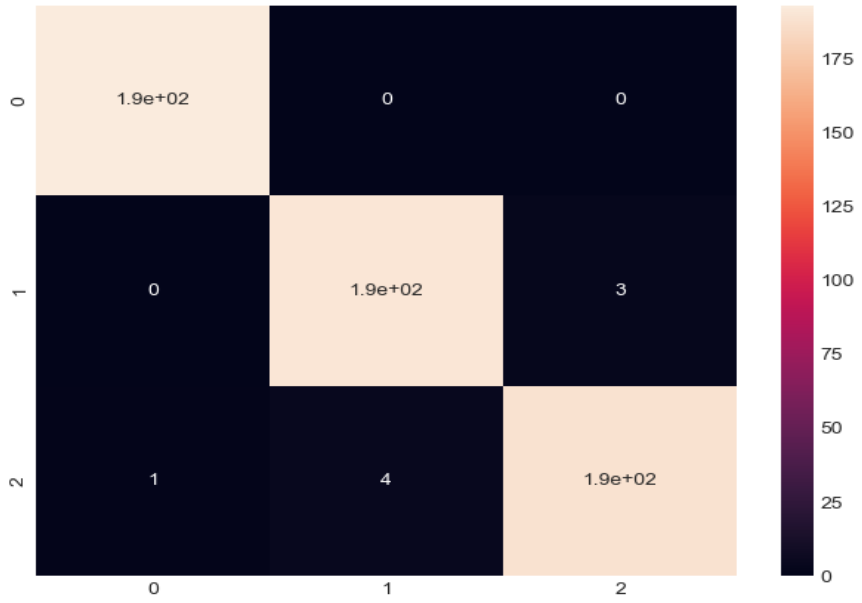

Figure 18: Confusion matrix

4 EDA and Visualisation

1. Exploratory Data Analysis and Visualisation was performed both Python and Power BI. In python, libraries like seaborn and matplotlib was used for implementing EDA.
2. Below Figure 19 depicts geographical map with Tweets count for all countries.

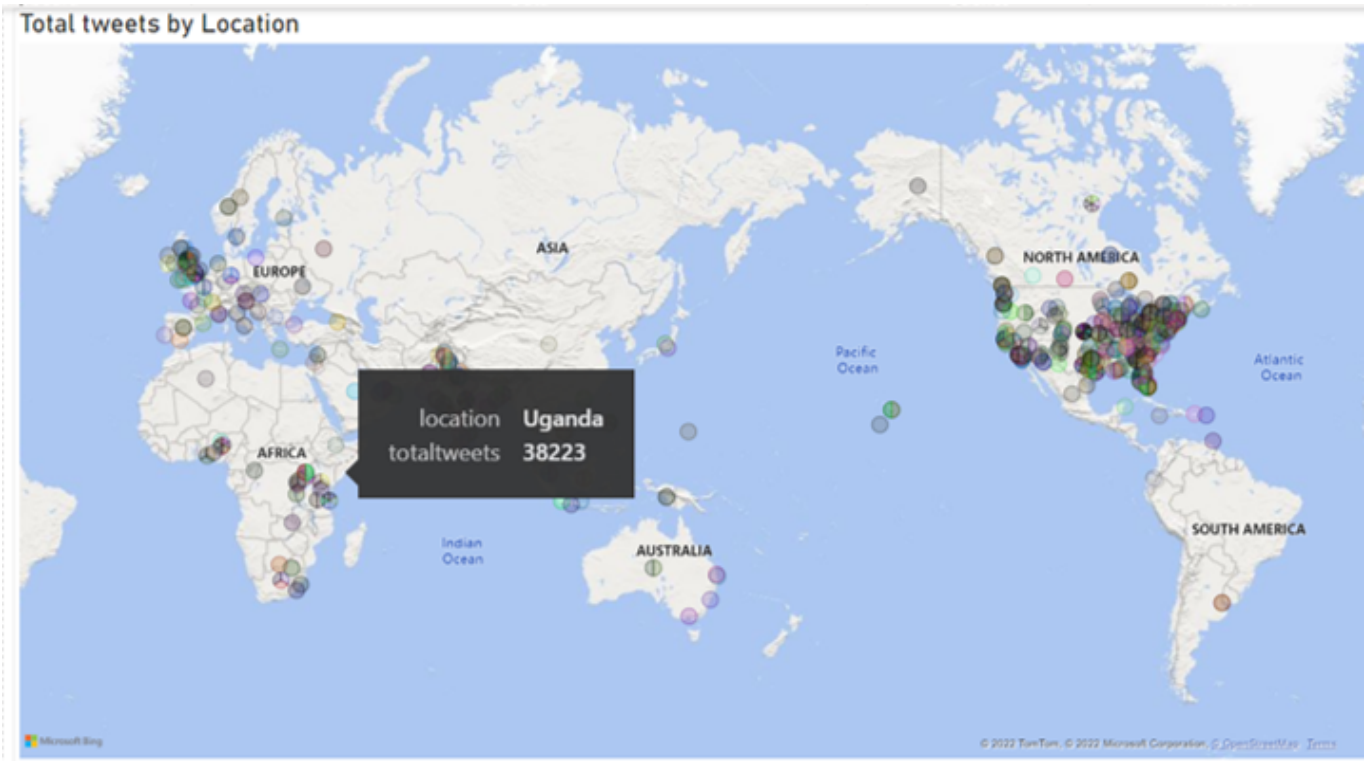

Figure 19: Total tweets by Country

3. Below Figure 20, depicts bar chart consisting of Top 10 locations with most follwers and total tweets. As we can clearly show, Kampala, Uganda has highest number of followers.

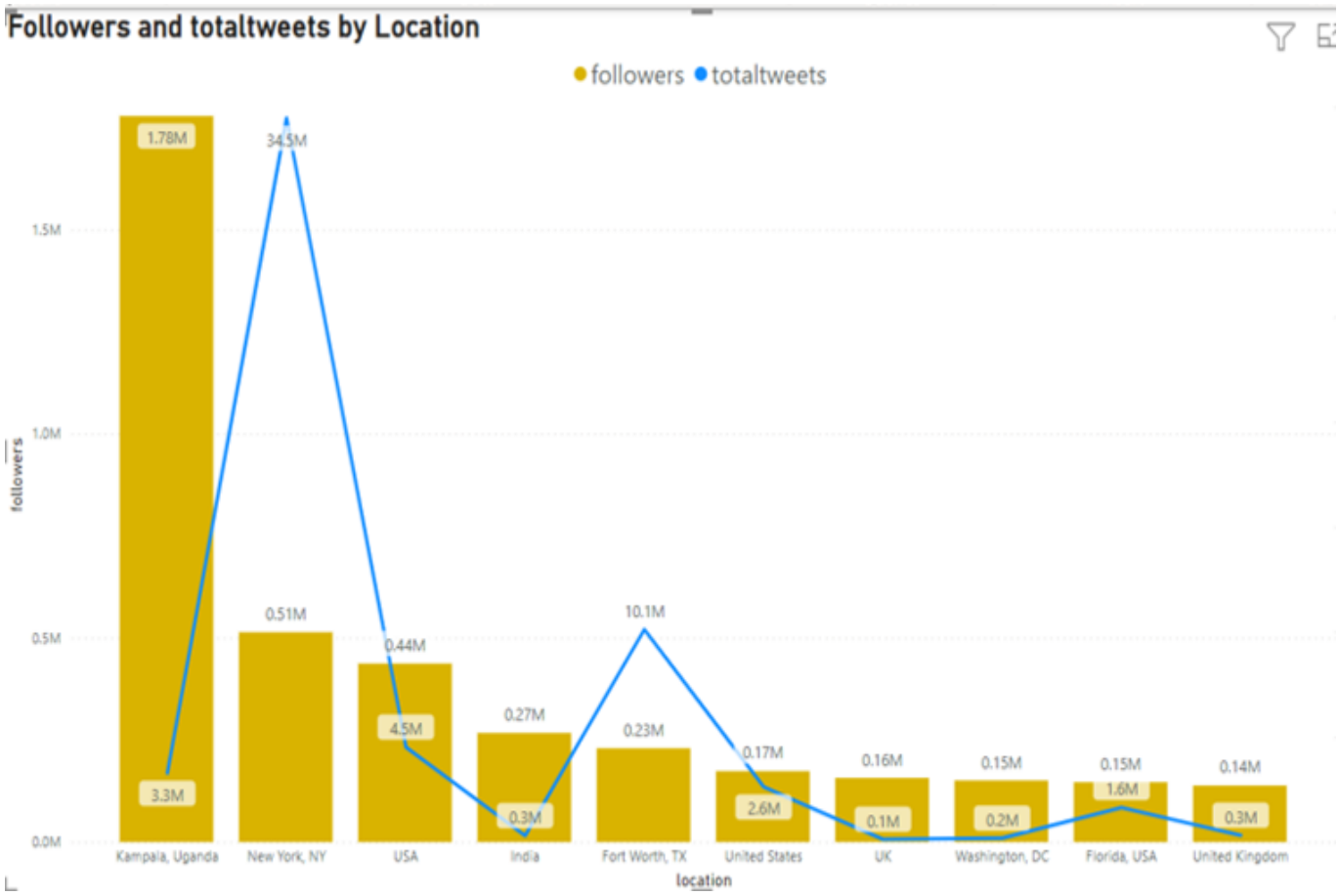

Figure 20: Top 10 Followers and TotalTweets by Location