SENTIMENT PREDICTOR FOR BRAND OWNERS.

Business Understanding

A brand name or identity is now considered among the most valuable asset for a going concern. According to Wikipedia , A brand is a name, term, design, symbol or any other feature that distinguishes one seller's good or service from those of other sellers.Brands are used in business, marketing, and advertising for recognition and, importantly, to create and store value as brand equity for the object identified, to the benefit of the brand's customers, its owners and shareholders. Consequently, in is important to business owners or brand owners to know and understand how their current and potential perceive heir brand. This information would feed into their strategy on how to enhance, protect, course correct, where applicable on the status of their brand.

Business Problem

Brand perception by current and would customer is key to a business success. A negative brand perception, especially, in this current age of social media and interconnectedness, can quickly wipe out the value of a company within a short time. And conversely, a positive perception can quickly add value to a company. Therefore, being able to gauge how people feel and perceive about one’s brand is a great asset.

Objective

To develop sentiment prdiction model based upon Natural language multiclassification with features as customer reviews and social media views.

Data Understanding

Our data is a CSV file of 9,093 records and 3 columns. The columns are tweet_texts, emotion_in_tweet_is_directed_at and is_there_an_emotion_directed_at_a brand_or_product. Essentially, the columns represent tweets of "customers" sentiments towards certain brands and/or products. We will apply data preparation techniques on out sentiment tweets in order to extract our features for modelling.

Model Summary

Business and data understanding.

Our data comprises customer social media posts on various products and services. This is invaluable information to a brand owners and businesses for them understand the opinions of customers towards their products. To be able to extract these opinions we will utilise Natural Language Processing to extract features with which to predict customer sentiments.

Data preparation

Our data comprises 9,093 records and three columns and our first action would be to extract a sizeable sample to work with. Our target ('y') under the "is_there_an_emotion_directed_at_a_brand_or_product " are texts and cannot be passed through the model we intend to use. We therefore assign integers to the four sentiments under the column. Before embarking on further data preparation steps, we will under separation of the data into test and train data sets using test_train_split. The following data preparation steps will under take on the train data set to enable us run abase model. -standardizing text by applying lower case function to the text from which we intend to extract our features. -tokenization of text by applying the tokenize library. This splits our text into individual tokens. This enables vectorization and modelling. In order to further improve the modelling process, we will remove stop words. Words which are too frequent in the tokenized data and carry little sentiment meaning will also be included into the stop words list for elimination.

Modelling

We have chosen to use the Multinomial Naive Bayes model for this project because the model is suited for text classification problems and easy to use. In addition, it is easy to implement, efficient with large data sets, has low computational cost and works for both binary and multiclass classifications. We have also chosen the TfidVectorizer as our vectorizer because it enables us not to rely the raw frequencies of word occurrences through scaling down the impact of token that occur more frequently. Emphasis is placed on the importance of tokens.

Evaluation

During the modelling process we under took two iterations after the base model. The base model had a cross validation score of 61%. We eliminated stop words under the first iteration and our score increased to 64%. However, when we tried to lemmatize the token under the second iteration the score declined to 61%. We therefore chose the first iteration as the best model. After passing through the test data through the chosen model we had an accuracy score of 64.8%. We also set up a confusion of which it was evident the model did not classify two labels quite well. The further steps would be to look into why these labels were not well classified by the model.

Data Preparation

Import of the necessary libraries.

In [2]:
```
import pandas as pd
import nltk
from nltk.probability import FreqDist
from nltk.corpus import stopwords
from nltk.tokenize import regexp_tokenize,word_tokenize,RegexpTokenizer
import matplotlib.pyplot as plt
import string
import re
```

In [3]:
```
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\asaav\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[3]: True

Loading and display of the data.

In [4]:
```
tweet=pd.read_csv('data/judge-1377884607_tweet_product_company.csv',encoding 
tweet.shape
```

Out[4]: (9093, 3)

In [5]:
```
tweet.head(11)
```

Out[5]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_branc
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Neg
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Po
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Po
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Neg
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Po
5	@teachntech00 New iPad Apps For #SpeechTherapy...	NaN	No emotion toward bra
6	NaN	NaN	No emotion toward bra
7	#SXSW is just starting, #CTIA is around the co...	Android	Po
8	Beautifully smart and simple idea RT @madebyma...	iPad or iPhone App	Po
9	Counting down the days to #sxsw plus strong Ca...	Apple	Po
10	Excited to meet the @samsungmobileus at #sxsw ...	Android	Po

The data comprises text data on tweets in form on text, the product and/or brand addressed in the tweet and the emotion toward that particular brand or product.Our focus will be on the first and the third column , whereby we intend to extract features from columns 1 which will predict the sentiment under column three.

Null Values Check

In [6]:
```
tweet.isna().sum()
```

Out[6]:
```
tweet_text                                             1
emotion_in_tweet_is_directed_at                     5802
is_there_an_emotion_directed_at_a_brand_or_product     0
dtype: int64
```

A quick check on whether the data contains null value in our columns of focus.The column tweet_text column has one null value while the last column has none. Null values would negatively impact our modelling and evaluation processes.

Generate a Sample

In [7]:
```
tweet_sample=tweet.sample(1000,random_state=22)
tweet_sample.head()
```

Out[7]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_p
845	My people listening to #google #sxsw @mention ...	NaN	No emotion toward brand or
2623	Whoa! line at the pop up apple store in downto...	iPad	Positive
8203	Find me at #SXSWi for info on how to get your ...	NaN	No emotion toward brand or
7430	If Apple pops up a store for an event like SXS...	NaN	No emotion toward brand or
3382	#technology #Apple saves #SXSW, set to open po...	NaN	Positive

The data is quite large. In order to optimize resources and runtime we select a random sample of 1000 records.

Assigning integers to target sentiments.

In [8]:
```
tweet_sample['is_there_an_emotion_directed_at_a_brand_or_product'].value_coun
```

Out[8]:
```
No emotion toward brand or product    602
Positive emotion                      326
Negative emotion                       58
I can't tell                           14
Name: is_there_an_emotion_directed_at_a_brand_or_product, dtype: int64
```

In [9]:
```
tweet_sample['is_there_an_emotion_directed_at_a_brand_or_product'].replace(to_
```

In [10]:
```
tweet_sample['is_there_an_emotion_directed_at_a_brand_or_product'].replace(to_
```

In [11]:
```
tweet_sample['is_there_an_emotion_directed_at_a_brand_or_product'].replace(to_
```

In [12]:
```
tweet_sample['is_there_an_emotion_directed_at_a_brand_or_product'].replace(to_
```

In [13]:
```
tweet_sample.head(10)
```

Out[13]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_
845	My people listening to #google #sxsw @mention ...	NaN	
2623	Whoa! line at the pop up apple store in downto...	iPad	
8203	Find me at #SXSWi for info on how to get your ...	NaN	
7430	If Apple pops up a store for an event like SXS...	NaN	
3382	#technology #Apple saves #SXSW, set to open po...	NaN	
2266	The next big thing? Hmmm. RT @mention Google t...	Google	
7875	#Posterous Joins The #SXSW Pile On With Poster...	NaN	
8656	So grateful my Twitterstream is mostly full of...	NaN	
1946	I was finally forced to google #sxsw. Why can'...	NaN	
1995	Was he standing? Talented... RT @mention eww &...	NaN	

To facilitate modelling we have assigned integers to the text sentiments under the target column is_there_an_emotion_directed_at_a_brand_or_product.

Train test split

In [14]:
```
X=tweet_sample[['tweet_text','emotion_in_tweet_is_directed_at']]
X
```

Out[14]:

	tweet_text	emotion_in_tweet_is_directed_at
845	My people listening to #google #sxsw @mention ...	NaN
2623	Whoa! line at the pop up apple store in downto...	iPad
8203	Find me at #SXSWi for info on how to get your ...	NaN
7430	If Apple pops up a store for an event like SXS...	NaN
3382	#technology #Apple saves #SXSW, set to open po...	NaN
...	...	...
7210	Privacy Could Headline Google Circles Social N...	NaN
3529	So I bought an iPad on impulse! Must be someth...	iPad
2436	Google and ACLU are buddies?ÛÏ@mention Google...	NaN
2865	Google prefers to launch hyped new Social feat...	Google
8196	Friends at #sxsw, can you take some 360 views ...	NaN

1000 rows × 2 columns

We have defined as the first two columns Tweet_tex and emotion_in_tweet_is_directed_at. However, our features will be extracted from the column tweet_ text in subsequent data cleaning processes of standardizing case, tokenization and vectorization.

In [15]:
```
y=tweet_sample['is_there_an_emotion_directed_at_a_brand_or_product']
y
```

Out[15]:
```
845     2
2623    1
8203    2
7430    2
3382    1
       ..
7210    2
3529    1
2436    2
2865    0
8196    1
Name: is_there_an_emotion_directed_at_a_brand_or_product, Length: 1000, dtyp
e: int64
```

We had mapped the text emotions under column "is_there_an_emotion_directed_at_a_brand_or_product" which is our target.

In [16]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.25,random_sta
```

Conduct a train test split with a 25% of the sample allocated to the test set.

In [17]:
```
X_train.isna().sum()
```

Out[17]:
```
tweet_text                          0
emotion_in_tweet_is_directed_at   488
dtype: int64
```

A quick check for null values under X_train.

In [18]:
```
X_train['tweet_text']=X_train['tweet_text'].fillna('').apply(str)
X_train
```

```
<ipython-input-18-be96715989f3>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_train['tweet_text']=X_train['tweet_text'].fillna('').apply(str)
```

Out[18]:

	tweet_text	emotion_in_tweet_is_directed_at
8017	Scored a signed print of the Jules Verne Googl...	NaN
960	.@mention on #sxsw with a Cr48. there's so muc...	NaN
1025	#technews Privacy Could Headline Google Circle...	NaN
1505	like that's bad RT @mention Sitting at a bar I...	NaN
615	#Apple to Hawk iPad 2 at #SXSW Festival Popup ...	iPad or iPhone App
...	...	...
4690	There is no bigger gathering of web-browsing, ...	NaN
6686	RT @mention Startups at #SXSW, @mention is giv...	NaN
3073	Brutal question served up to Marissa Mayer abo...	NaN
7487	Google Maps Mobile Route Around Traffic featur...	NaN
8506	(via @mention #SXSW 2011: The #Google and #Bin...	NaN

750 rows × 2 columns

In [19]:
```
X_train.isna().sum()
```

Out[19]:
```
tweet_text                            0
emotion_in_tweet_is_directed_at     488
dtype: int64
```

In [20]:
```
y_train.value_counts()
```

Out[20]:
```
2    458
1    252
0     40
Name: is_there_an_emotion_directed_at_a_brand_or_product, dtype: int64
```

Standardize text

In [21]:
```
X_train['tweet_text']=X_train['tweet_text'].str.lower()
X_train
```

```
<ipython-input-21-3ac2e5a0ee27>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_train['tweet_text']=X_train['tweet_text'].str.lower()
```

Out[21]:

	tweet_text	emotion_in_tweet_is_directed_at
8017	scored a signed print of the jules verne googl...	NaN
960	.@mention on #sxsw with a cr48. there's so muc...	NaN
1025	#technews privacy could headline google circle...	NaN
1505	like that's bad rt @mention sitting at a bar l...	NaN
615	#apple to hawk ipad 2 at #sxsw festival popup ...	iPad or iPhone App
...	...	...
4690	there is no bigger gathering of web-browsing, ...	NaN
6686	rt @mention startups at #sxsw, @mention is giv...	NaN
3073	brutal question served up to marissa mayer abo...	NaN
7487	google maps mobile route around traffic featur...	NaN
8506	(via @mention #sxsw 2011: the #google and #bin...	NaN

750 rows × 2 columns

We standardize case for the column "tweet_text" to avoid the same word being considered as different based on the case of its letters.

Tokenizing

In [22]:
```
pattern= r"(?u)\b\w\w+\b"
tokenizer=RegexpTokenizer(pattern)
```

In [23]:
```
X_train['tweet_text_token']=X_train['tweet_text'].apply(tokenizer.tokenize)
X_train
```

```
<ipython-input-23-e56a9f61ce29>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_train['tweet_text_token']=X_train['tweet_text'].apply(tokenizer.tokenize)
```

Out[23]:

	tweet_text	emotion_in_tweet_is_directed_at	tweet_text_token
8017	scored a signed print of the jules verne googl...	NaN	[scored, signed, print, of, the, jules, verne,...
960	.@mention on #sxsw with a cr48. there's so muc...	NaN	[mention, on, sxsw, with, cr48, there, so, muc...
1025	#technews privacy could headline google circle...	NaN	[technews, privacy, could, headline, google, c...
1505	like that's bad rt @mention sitting at a bar l...	NaN	[like, that, bad, rt, mention, sitting, at, ba...
615	#apple to hawk ipad 2 at #sxsw festival popup ...	iPad or iPhone App	[apple, to, hawk, ipad, at, sxsw, festival, po...
...	...	...	...
4690	there is no bigger gathering of web-browsing, ...	NaN	[there, is, no, bigger, gathering, of, web, br...
6686	rt @mention startups at #sxsw, @mention is giv...	NaN	[rt, mention, startups, at, sxsw, mention, is,...
3073	brutal question served up to marissa mayer abo...	NaN	[brutal, question, served, up, to, marissa, ma...
7487	google maps mobile route around traffic featur...	NaN	[google, maps, mobile, route, around, traffic,...
8506	(via @mention #sxsw 2011: the #google and #bin...	NaN	[via, mention, sxsw, 2011, the, google, and, b...

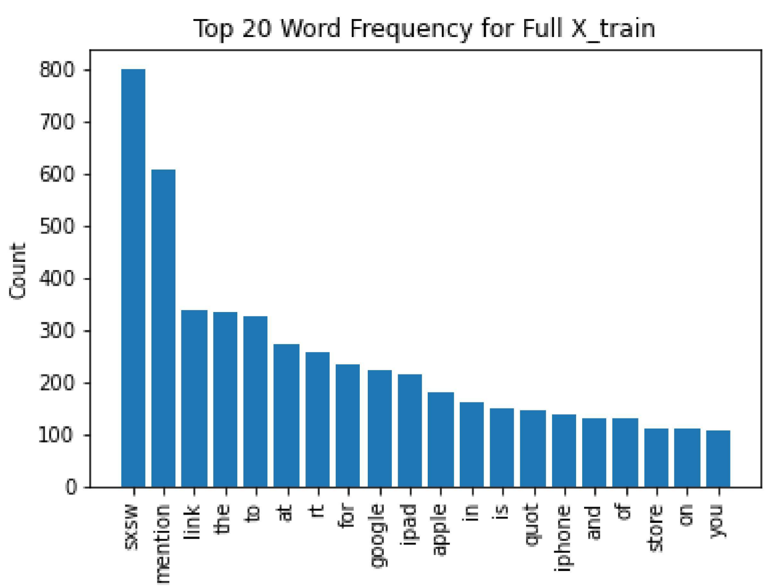
750 rows × 3 columns

We now tokenize the "tweet_column" , which is splitting the text into tokens which would enable us extract features.

Exploratory Data Analysis.

In [24]:
```
from matplotlib.ticker import MaxNLocator
def visualize_top_20(freq_dist, title):

    # Extract data for plotting
    top_20 = list(zip(*freq_dist.most_common(20)))
    tokens = top_20[0]
    counts = top_20[1]

    # Set up plot and plot data
    fig, ax = plt.subplots()
    ax.bar(tokens, counts)

    # Customize plot appearance
    ax.set_title(title)
    ax.set_ylabel("Count")
    ax.yaxis.set_major_locator(MaxNLocator(integer=True))
    ax.tick_params(axis="x", rotation=90)
```

In [25]:
```
train_freq_dist = FreqDist(X_train['tweet_text_token'].explode())
visualize_top_20(train_freq_dist, "Top 20 Word Frequency for Full X_train")
```



After tokenizing, we seek to find out what are these words in the text and how frequent are they. We generate a frequency distribution using the FreqDist library. This will inform our decision making in our parameter tuning.

Baseline Model

We now create a base model based upon our non-tuned features to have a baseline for our subsequent models based upon tuned features. We will utilise the TfidVectorizer and the MultiNomialNB classifier to generate this and subsequent models. The TfidVectorizer enables us not to rely the raw frequencies of word occurrences through scaling down the impact of token that occur more frequently. Emphasis is placed on the importance of tokens. The multinomial NB classifier is ideal for text classification problems and easy to use. It assumes that the value of a particular feature is independent of the value of any other feature, given the class variable. It is easy to implement, efficient with large data sets, has low computational cost and works for both binary and multiclass classifications.

In [26]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
Tfid = TfidfVectorizer(max_features=20)
X_train_vectorized = Tfid.fit_transform(X_train['tweet_text'])
pd.DataFrame.sparse.from_spmatrix(X_train_vectorized, columns=Tfid.get_featur
```

Out[26]:

	and	apple	at	for	google	in	ipad	iphone	is	
0	0.000000	0.000000	0.000000	0.000000	0.462913	0.000000	0.00000	0.000000	0.000000	C
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	C
2	0.000000	0.000000	0.000000	0.000000	0.740153	0.000000	0.00000	0.000000	0.000000	C
3	0.000000	0.000000	0.365292	0.000000	0.000000	0.000000	0.00000	0.470298	0.000000	C
4	0.000000	0.427992	0.351911	0.000000	0.000000	0.000000	0.38281	0.000000	0.000000	C
...	...	...	...	...	...	...	...	...	...	
745	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.519814	0.515627	C
746	0.360993	0.000000	0.269581	0.290798	0.000000	0.000000	0.00000	0.000000	0.344278	C
747	0.000000	0.000000	0.000000	0.453825	0.000000	0.000000	0.00000	0.000000	0.000000	C
748	0.567987	0.000000	0.000000	0.000000	0.461401	0.537383	0.00000	0.000000	0.000000	C
749	0.504288	0.000000	0.000000	0.000000	0.409656	0.477116	0.00000	0.000000	0.000000	C

750 rows × 20 columns

In [27]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import cross_val_score
baseline_model = MultinomialNB()
baseline_cv = cross_val_score(baseline_model, X_train_vectorized, y_train)
baseline_cv
```

Out[27]:
```
array([0.61333333, 0.61333333, 0.61333333, 0.60666667, 0.60666667])
```

The base model has accuracy score ranging from 60% to 61%. Meaning 61% of the time the model will classify correctly. Cross validation chosen as an evaluation method because it is a robust measure of a model’s predictive performance and it allows for efficient use of data.We will go through further preprocessing steps in order to improve the model's performance.

Further Preprocessing to improve the model.

Removal of stop words.

In [28]:
```
stopwords_list = stopwords.words('english')
len(stopwords_list)
```

Out[28]: 179

In the frequency table above, there are frequent words which are not part of the stop word list yet they do not aid in arriving at someone's opinion. These words have been added to the stop words list.

In [29]:
```
newstopwords=['sxsw','ipad','apple','google','iphone','android','rt']
stopwords_list.extend(newstopwords)
len(stopwords_list)
```

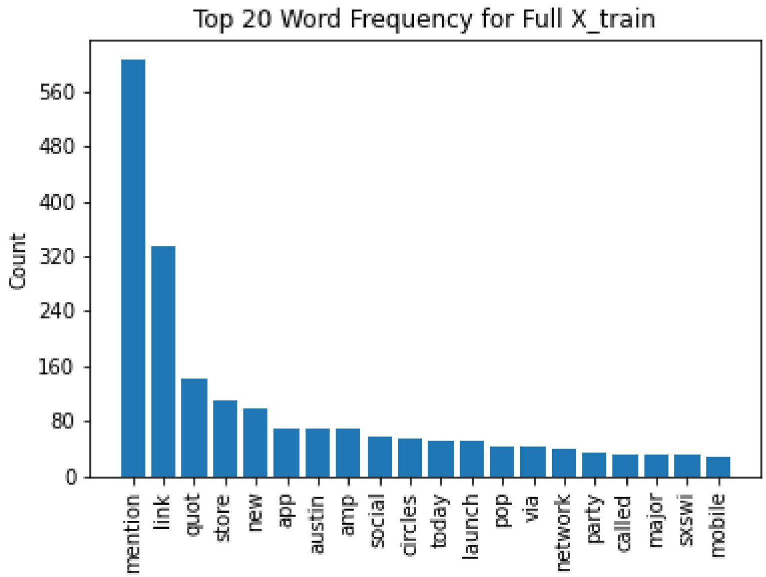
Out[29]: 186

In [30]:
```
def remove_stopwords(token_list):

    stopwords_removed = [token for token in token_list if token not in stopwo
    return stopwords_removed
```

In [31]:
```
X_train['tweet_text_without_stopwords'] = X_train['tweet_text_token'].apply(r
```

```
<ipython-input-31-e324abb22107>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  X_train['tweet_text_without_stopwords'] = X_train['tweet_text_token'].appl
y(remove_stopwords)
```

In [32]:
```
train_freq_dist = FreqDist(X_train['tweet_text_without_stopwords'].explode())
visualize_top_20(train_freq_dist, "Top 20 Word Frequency for Full X_train")
```



In [33]:
```
tfidf1 = TfidfVectorizer(
    max_features=20,
    stop_words=stopwords_list)
X_train_vectorized1 = tfidf1.fit_transform(X_train["tweet_text"])
pd.DataFrame.sparse.from_spmatrix(X_train_vectorized1, columns=tfidf1.get_fea
```

Out[33]:

	amp	app	austin	called	circles	launch	link	major	mention	mobile	net
0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.752369	0.0	0.658742	0.000000	
1	0.0	0.000000	0.0	0.0	0.000000	0.0	0.752369	0.0	0.658742	0.000000	
2	0.0	0.000000	0.0	0.0	0.667513	0.0	0.336578	0.0	0.000000	0.000000	
3	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	1.000000	0.000000	
4	0.0	0.000000	0.0	0.0	0.000000	0.0	0.479360	0.0	0.419707	0.000000	
...	...	...	...	...	...	...	...	...	...	...	
745	0.0	0.882346	0.0	0.0	0.000000	0.0	0.470602	0.0	0.000000	0.000000	
746	0.0	0.000000	0.0	0.0	0.000000	0.0	0.355797	0.0	0.934563	0.000000	
747	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.0	1.000000	0.000000	
748	0.0	0.000000	0.0	0.0	0.000000	0.0	0.385384	0.0	0.000000	0.922756	
749	0.0	0.000000	0.0	0.0	0.000000	0.0	0.399879	0.0	0.350117	0.000000	

750 rows × 20 columns

In [34]:
```
stopwords_removed_cv = cross_val_score(baseline_model, X_train_vectorized1, y
stopwords_removed_cv
```

Out[34]:
```
array([0.64      , 0.62      , 0.58      , 0.6       , 0.62666667])
```

After removing the stop words the accuracy score now ranges between 64% to 58%. The is an improvement on the base model. Let us try some more preprocessing and try and improve the score. Let us try lemmatization.

Lemmatize to improve model

In [35]:
```
from nltk.stem import WordNetLemmatizer
Lemmatizer=WordNetLemmatizer()

def lem_and_tokenize(document):
    tokens = tokenizer.tokenize(document)
    return [Lemmatizer.lemmatize(token) for token in tokens]
```

In [36]:
```
# Lemmatize stop words
Lemmed_stopwords = [Lemmatizer.lemmatize(word) for word in stopwords_list]
```

In [37]:
```
tfidf2 = TfidfVectorizer(
    max_features=20,
    stop_words=Lemmed_stopwords,
    tokenizer=lem_and_tokenize
)
X_train_vectorized2 = tfidf2.fit_transform(X_train["tweet_text"])
pd.DataFrame.sparse.from_spmatrix(X_train_vectorized2, columns=tfidf2.get_fea
```

Out[37]:

	amp	app	austin	called	circle	get	launch	line	link	major	mention	netw
0	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	0.751826	0.0	0.659362	
1	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	0.751826	0.0	0.659362	
2	0.0	0.00000	0.0	0.0	0.667638	0.0	0.0	0.0	0.336082	0.0	0.000000	
3	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	1.000000	
4	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	0.480517	0.0	0.421420	
...	...	...	...	...	...	...	...	...	...	...	...	
745	0.0	0.88267	0.0	0.0	0.000000	0.0	0.0	0.0	0.469993	0.0	0.000000	
746	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	0.355281	0.0	0.934759	
747	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	1.000000	
748	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	1.000000	0.0	0.000000	
749	0.0	0.00000	0.0	0.0	0.000000	0.0	0.0	0.0	0.399321	0.0	0.350210	

750 rows × 20 columns

In [38]:
```
stopwords_removed_cv_lem = cross_val_score(baseline_model, X_train_vectorized:
stopwords_removed_cv_lem
```

Out[38]:
```
array([0.61333333, 0.60666667, 0.58666667, 0.60666667, 0.60666667])
```

After lemmatization the accuracy score ranges between 61% to 58.6%. The is worse than the previous model but almost a par as the base model. We will then choose the second model (i.e. model after removing stop words) as the final model.

Final Model

In [39]:
```
final_model = MultinomialNB()

final_model.fit(X_train_vectorized1, y_train)
final_model.score(X_train_vectorized1, y_train)
```

Out[39]:
```
0.624
```

In [40]:
```
X_test.isna().sum()
```

Out[40]:
```
tweet_text                            0
emotion_in_tweet_is_directed_at     163
dtype: int64
```

In [41]:
```
X_test['tweet_text']=X_test['tweet_text'].fillna('').apply(str)
X_test
```

```
<ipython-input-41-aebd2e1d69ab>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X_test['tweet_text']=X_test['tweet_text'].fillna('').apply(str)
```

Out[41]:

	tweet_text	emotion_in_tweet_is_directed_at
8516	From @mention Marissa Mayer: 40% Of Google Map...	Other Google product or service
8167	#madmen RT @mention Hanging out with @mention ...	NaN
59	@mention @mention & @mention having fun ...	NaN
8037	Apple is so smart: The iPad 2 Takes Over #SXSW...	Apple
2744	every time u hold yur ipad 2 up in the air to ...	NaN
...	...	...
4711	There is nothing quite like #SXSW to make you ...	iPad
4890	Found a road dawg to check out this iPad store...	iPad
975	#Apple to Open Pop-Up Shop at #SXSW [REPORT]: ...	NaN
4874	Be sure to stop by our SXSW booth today. Show ...	iPad
7507	Apple is quarter of the music industry and 70%...	Apple

250 rows × 2 columns

In [42]:
```
y_test.value_counts()
```

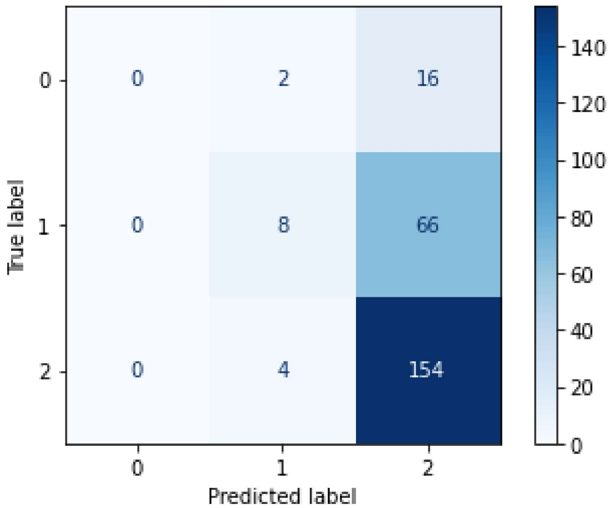
Out[42]:
```
2    158
1     74
0     18
Name: is_there_an_emotion_directed_at_a_brand_or_product, dtype: int64
```

In [43]:
```
X_test_vectorized = tfidf1.transform(X_test['tweet_text'])
```

In [44]:
```
final_model.score(X_test_vectorized,y_test)
```

Out[44]:
```
0.648
```

In [46]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

cnf_matrix = confusion_matrix(y_test, final_model.predict(X_test_vectorized))
disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix, display_labels=fin
disp.plot(cmap=plt.cm.Blues);
```



In terms of accuracy the moel has performed well with a score of 64.8%

Considering the confusion marix the model has correctly classified sentiment 2 (no emotion towards brand or product), has not correctly classified sentiment 0 (negative emotion)and sentiment 1 (positive emotion.) We should now examine the misplaced sentiments 0 and 1 to detemine what may have caused their misclassification.Is additional feature engineering required or preprocessing.

Type *Markdown* and LaTeX: α^2