## Import Packages and Libraries

In [1]:

```
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf
from tensorflow.keras import layers
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
import numpy as np
import pandas as pd
import re
import gensim
from gensim import corpora
from gensim import similarities
from gensim import models
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer, TfidfVe
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, f1
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB, BernoulliNB, GaussianNB
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import *
import warnings
warnings.filterwarnings("ignore")
from datetime import datetime
```

```
Using TensorFlow backend.
```

## Import Data File and Cleaning

In [3]:

```
# data_file = "SHOPEE_MAYBELLINE_CLEAN_V2.csv"
# data_file = "Lazada_sentiment.csv"
data_file = "Shopee_AllData_Sentiment_v2.csv"
data = pd.read_csv(data_file)
data.columns = data.columns.str.strip().str.replace(" ","_")
# data.info()
# data.head()

# data.drop(columns=['Brand','Category','Product_Name','Price','Reviewer','Product_Purc
# review_list = data['Review'].tolist()
# polarity_list = data['Polarity'].tolist()

reviews = data['Review']
# polarity = data['Polarity']
# print (reviews)

review_docs = []
for each_reviews in reviews:
    temp = each_reviews.split(" ")
    review_docs.append(temp)
# print (review_docs)

# Make sure all words are in lowercase
reviews_lower = [[each_word.lower() for each_word in each_review] for each_review in re
# print (reviews_lower)

# Use regular expressions to keep only allphabetical words
reviews_alpha = [[each_word for each_word in each_review if re.search('^[a-z]+$', each_
# print (reviews_alpha)

# Remove stop words
stop_list = stopwords.words('english')
reviews_stop = [[each_word for each_word in each_review if each_word not in stop_list]
# print (reviews_stop)

# Porter Stemming
stemmer = PorterStemmer()
reviews_stem = [[stemmer.stem(each_word) for each_word in each_review] for each_review
# print (reviews_stem)

all_data_cleaned = []
for each_sentence in reviews_stem:
    sentence = ""
    for each_word in each_sentence:
        sentence += each_word + " "
    sentence = sentence[0:-1]
    all_data_cleaned.append(sentence)
# print (all_data_cleaned)

polarity_raw = data['Polarity']
polarity_0_and_1 = []
for each_polarity in polarity_raw:
    if int(each_polarity) == int("0"):
        polarity_0_and_1.append(0.5)
    if int(each_polarity) == int("-1"):
        polarity_0_and_1.append(int(0))
    if int(each_polarity) == int("1"):
        polarity_0_and_1.append(int(1))
# print (polarity)

# reviews = all_data_cleaned
# print (len(reviews))

# product_purchase = data['Product_Purchase']
# print (product_purchase[0:10])



```

## Building a Model - Count Vector

1. Multinomial NB
2. Bernoulli NB
3. Logistic Regression
4. Support Vector Machine

In [4]:

```python
### done
print (datetime.now())

Classifiers = [MultinomialNB(), BernoulliNB(), LogisticRegression(), SVC()]

reviews = all_data_cleaned
polarity = data['Polarity']
X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,

countVectorizer = CountVectorizer(min_df = 4, max_df=0.85)
X_train = countVectorizer.fit_transform(X_train)
X_test = countVectorizer.transform(X_test)

for i in range(len(Classifiers)):
    clf = Classifiers[i]
    clf_name = "Test"

    if i == int(0):
        clf_name = "Multinomial Naive Bayes"
        clf = Classifiers[i]
    elif i == int(1):
        clf_name = "Bernoulli Naive Bayes"
        clf = Classifiers[i]
    elif i == int(2):
        clf_name = "Logistic Regression"
        clf = Classifiers[i]
    elif i == int(3):
        clf_name = "Support Vector Machine"
        clf = Classifiers[i]
#         parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
        parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rb
        clf = GridSearchCV(estimator = clf, param_grid = parameters)

    clf.fit(X_train, y_train)
    clf_ypred = clf.predict(X_test)
    f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
    accuracy_clf = accuracy_score(y_test, clf_ypred)
    print ("F1-score of", clf_name, "with Count Vector is:", f1_clf*100)
    print ("Accuracy of", clf_name, "with Count Vector is:", accuracy_clf*100)

    if clf_name == "Support Vector Machine":
        print (clf.best_params_)

print (datetime.now())
```

```
2020-03-17 20:10:57.739423
F1-score of Multinomial Naive Bayes with Count Vector is: 76.19581530650309
Accuracy of Multinomial Naive Bayes with Count Vector is: 75.9212991880075
F1-score of Bernoulli Naive Bayes with Count Vector is: 71.70435920572501
Accuracy of Bernoulli Naive Bayes with Count Vector is: 70.70580886945659
F1-score of Logistic Regression with Count Vector is: 76.42136878939021
Accuracy of Logistic Regression with Count Vector is: 76.57713928794504
F1-score of Support Vector Machine with Count Vector is: 76.90244294161393
Accuracy of Support Vector Machine with Count Vector is: 77.29544034978139
{'C': 3, 'degree': 1, 'gamma': 0.01, 'kernel': 'poly'}
2020-03-17 20:23:00.977783
```

## Building a Model - TFIDF (use_idf = False)

1. Multinomial NB
2. Bernoulli NB

3. Logistic Regression
4. Support Vector Machine

In [5]:

```python
### done
print (datetime.now())

Classifiers = [MultinomialNB(), BernoulliNB(), LogisticRegression(), SVC()]

reviews = all_data_cleaned
polarity = data['Polarity']
X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,

tf_Vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df=0.85)
X_train = tf_Vectorizer.fit_transform(X_train)
X_test = tf_Vectorizer.transform(X_test)

for i in range(len(Classifiers)):
    clf = Classifiers[i]
    clf_name = "Test"

    if i == int(0):
        clf_name = "Multinomial Naive Bayes"
        clf = Classifiers[i]
    elif i == int(1):
        clf_name = "Bernoulli Naive Bayes"
        clf = Classifiers[i]
    elif i == int(2):
        clf_name = "Logistic Regression"
        clf = Classifiers[i]
    elif i == int(3):
        clf_name = "Support Vector Machine"
        clf = Classifiers[i]
#         parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001], '
        parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rb
        clf = GridSearchCV(estimator = clf, param_grid = parameters)

    clf.fit(X_train, y_train)
    clf_ypred = clf.predict(X_test)
    f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
    accuracy_clf = accuracy_score(y_test, clf_ypred)
    print ("F1-score of", clf_name, "with TFIDF (use_idf = False) is:", f1_clf*100)
    print ("Accuracy of", clf_name, "with TFIDF (use_idf = False) is:", accuracy_clf*10

    if clf_name == "Support Vector Machine":
        print (clf.best_params_)

print (datetime.now())
```

```
2020-03-17 20:23:00.988728
F1-score of Multinomial Naive Bayes with TFIDF (use_idf = False) is: 75.88387291376574
Accuracy of Multinomial Naive Bayes with TFIDF (use_idf = False) is: 75.9525296689569
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = False) is: 71.70435920572501
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = False) is: 70.70580886945659
F1-score of Logistic Regression with TFIDF (use_idf = False) is: 77.17643970976897
Accuracy of Logistic Regression with TFIDF (use_idf = False) is: 77.67020612117427
F1-score of Support Vector Machine with TFIDF (use_idf = False) is: 77.30317954556676
Accuracy of Support Vector Machine with TFIDF (use_idf = False) is: 77.79512804497189
{'C': 3, 'degree': 1, 'gamma': 0.1, 'kernel': 'poly'}
2020-03-17 20:31:34.633966
```

## Building a Model - TFIDF (use_idf = True)

1. Multinomial NB
2. Bernoulli NB
3. Logistic Regression
4. Support Vector Machine

In [6]:

```
### done
print (datetime.now())

Classifiers = [MultinomialNB(), BernoulliNB(), LogisticRegression(), SVC()]

reviews = all_data_cleaned
polarity = data['Polarity']
X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,

tfidfVectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
X_train = tfidfVectorizer.fit_transform(X_train)
X_test = tfidfVectorizer.transform(X_test)

for i in range(len(Classifiers)):
    clf = Classifiers[i]
    clf_name = "Test"

    if i == int(0):
        clf_name = "Multinomial Naive Bayes"
        clf = Classifiers[i]
    elif i == int(1):
        clf_name = "Bernoulli Naive Bayes"
        clf = Classifiers[i]
    elif i == int(2):
        clf_name = "Logistic Regression"
        clf = Classifiers[i]
    elif i == int(3):
        clf_name = "Support Vector Machine"
        clf = Classifiers[i]
#         parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
        parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rb
        clf = GridSearchCV(estimator = clf, param_grid = parameters)

    clf.fit(X_train, y_train)
    clf_ypred = clf.predict(X_test)
    f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
    accuracy_clf = accuracy_score(y_test, clf_ypred)
    print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
    print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*10

    if clf_name == "Support Vector Machine":
        print (clf.best_params_)

print (datetime.now())
```

```
2020-03-17 20:31:34.647966
F1-score of Multinomial Naive Bayes with TFIDF (use_idf = True) is: 75.25011540044297
Accuracy of Multinomial Naive Bayes with TFIDF (use_idf = True) is: 75.26545908806995
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 71.70435920572501
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 70.70580886945659
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 77.07386881969714
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 77.57651467832605
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 77.32012381935355
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 77.82635852592131
{'C': 1, 'degree': 1, 'gamma': 0.1, 'kernel': 'poly'}
2020-03-17 20:40:18.277311
```

## Building a Model - PCA (n=2)

Multinomial Naive Bayes cannot do PCA as the input is negative

2. Bernoulli NB
3. Logistic Regression
4. Support Vector Machine

In [7]:

```python
Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]

reviews = all_data_cleaned
polarity = data['Polarity']
X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,

# vectorizer = CountVectorizer()
# vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train.toarray())

df_train = pd.DataFrame(X_train)
df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
df_train.columns = ['pca_1', 'pca_2', 'target']
df_train['pca_1'].replace("", np.nan, inplace = True)
df_train['pca_2'].replace("", np.nan, inplace = True)
df_train['target'].replace("", np.nan, inplace = True)
df_train.dropna(subset=['pca_1', 'pca_2', 'target'], inplace = True)
df_train['pca_1'] = df_train['pca_1'].astype(float)
df_train['pca_2'] = df_train['pca_2'].astype(float)

X_test = pca.transform(X_test.toarray())
df_test = pd.DataFrame(X_test)
df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
df_test.columns = ['pca_1', 'pca_2', 'target']
df_test.describe(include='all')
df_test['pca_1'].replace("", np.nan, inplace = True)
df_test['pca_2'].replace("", np.nan, inplace = True)
df_test['target'].replace("", np.nan, inplace = True)
df_test.dropna(subset=['pca_1', 'pca_2', 'target'], inplace = True)
df_test['pca_1'] = df_test['pca_1'].astype(float)
df_test['pca_2'] = df_test['pca_2'].astype(float)

for i in range(len(Classifiers)):
    clf = Classifiers[i]
    clf_name = "Test"

    if i+1 == int(0):
        clf_name = "Multinomial Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(1):
        clf_name = "Bernoulli Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(2):
        clf_name = "Logistic Regression"
        clf = Classifiers[i]
    elif i+1 == int(3):
        clf_name = "Support Vector Machine"
        clf = Classifiers[i]
#         parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
        parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rb
        clf = GridSearchCV(estimator = clf, param_grid = parameters)

    clf.fit(X_train, y_train)
    clf_ypred = clf.predict(X_test)
    f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
    accuracy_clf = accuracy_score(y_test, clf_ypred)
    print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
    print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*10

    if clf_name == "Support Vector Machine":
        print (clf.best_params_)

```

```
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 61.06785170901428
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 61.242973141786386
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 72.22205235946649
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 72.51717676452218
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 68.98088648244212
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 69.92504684572143
{'C': 1, 'degree': 1, 'gamma': 0.1, 'kernel': 'linear'}
```

### Building a Model - PCA (n=3)

Multinomial Naive Bayes cannot do PCA as the input is negative

2. Bernoulli NB
3. Logistic Regression
4. Support Vector Machine

In [8]:

```python
Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]

reviews = all_data_cleaned
polarity = data['Polarity']
X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,

# vectorizer = CountVectorizer()
# vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

pca = PCA(n_components = 3)
X_train = pca.fit_transform(X_train.toarray())

df_train = pd.DataFrame(X_train)
df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
df_train.columns = ['pca_1', 'pca_2', 'pca_3', 'target']
df_train['pca_1'].replace("", np.nan, inplace = True)
df_train['pca_2'].replace("", np.nan, inplace = True)
df_train['pca_3'].replace("", np.nan, inplace = True)
df_train['target'].replace("", np.nan, inplace = True)
df_train.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'target'], inplace = True)
df_train['pca_1'] = df_train['pca_1'].astype(float)
df_train['pca_2'] = df_train['pca_2'].astype(float)
df_train['pca_3'] = df_train['pca_3'].astype(float)

X_test = pca.transform(X_test.toarray())
df_test = pd.DataFrame(X_test)
df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
df_test.columns = ['pca_1', 'pca_2', 'pca_3', 'target']
df_test.describe(include='all')
df_test['pca_1'].replace("", np.nan, inplace = True)
df_test['pca_2'].replace("", np.nan, inplace = True)
df_test['pca_3'].replace("", np.nan, inplace = True)
df_test['target'].replace("", np.nan, inplace = True)
df_test.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'target'], inplace = True)
df_test['pca_1'] = df_test['pca_1'].astype(float)
df_test['pca_2'] = df_test['pca_2'].astype(float)
df_test['pca_3'] = df_test['pca_3'].astype(float)

for i in range(len(Classifiers)):
    clf = Classifiers[i]
    clf_name = "Test"

    if i+1 == int(0):
        clf_name = "Multinomial Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(1):
        clf_name = "Bernoulli Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(2):
        clf_name = "Logistic Regression"
        clf = Classifiers[i]
    elif i+1 == int(3):
        clf_name = "Support Vector Machine"
        clf = Classifiers[i]
#         parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
        parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rb
        clf = GridSearchCV(estimator = clf, param_grid = parameters)

    clf.fit(X_train, y_train)
    clf_ypred = clf.predict(X_test)
    f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
    accuracy_clf = accuracy_score(y_test, clf_ypred)
    print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
    print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*10

    if clf_name == "Support Vector Machine":
        print (clf.best_params_)

```

```
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 64.36129718448363
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 64.49094316052467
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 73.48277660571046
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 73.89131792629607
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 72.6141315897205
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 73.36039975015616
{'C': 1, 'degree': 1, 'gamma': 0.1, 'kernel': 'linear'}
```

## Building a Model - PCA (n=4)

Multinomial Naive Bayes cannot do PCA as the input is negative

2. Bernoulli NB
3. Logistic Regression
4. Support Vector Machine

In [9]:

```
Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]

reviews = all_data_cleaned
polarity = data['Polarity']
X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,

# vectorizer = CountVectorizer()
# vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

pca = PCA(n_components = 4)
X_train = pca.fit_transform(X_train.toarray())

df_train = pd.DataFrame(X_train)
df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
df_train.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target']
df_train['pca_1'].replace("", np.nan, inplace = True)
df_train['pca_2'].replace("", np.nan, inplace = True)
df_train['pca_3'].replace("", np.nan, inplace = True)
df_train['pca_4'].replace("", np.nan, inplace = True)
df_train['target'].replace("", np.nan, inplace = True)
df_train.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target'], inplace = True)
df_train['pca_1'] = df_train['pca_1'].astype(float)
df_train['pca_2'] = df_train['pca_2'].astype(float)
df_train['pca_3'] = df_train['pca_3'].astype(float)
df_train['pca_4'] = df_train['pca_4'].astype(float)

X_test = pca.transform(X_test.toarray())
df_test = pd.DataFrame(X_test)
df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
df_test.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target']
df_test.describe(include='all')
df_test['pca_1'].replace("", np.nan, inplace = True)
df_test['pca_2'].replace("", np.nan, inplace = True)
df_test['pca_3'].replace("", np.nan, inplace = True)
df_test['pca_4'].replace("", np.nan, inplace = True)
df_test['target'].replace("", np.nan, inplace = True)
df_test.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target'], inplace = True)
df_test['pca_1'] = df_test['pca_1'].astype(float)
df_test['pca_2'] = df_test['pca_2'].astype(float)
df_test['pca_3'] = df_test['pca_3'].astype(float)
df_test['pca_4'] = df_test['pca_4'].astype(float)

for i in range(len(Classifiers)):
    clf = Classifiers[i]
    clf_name = "Test"

    if i+1 == int(0):
        clf_name = "Multinomial Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(1):
        clf_name = "Bernoulli Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(2):
        clf_name = "Logistic Regression"
        clf = Classifiers[i]
    elif i+1 == int(3):
        clf_name = "Support Vector Machine"
        clf = Classifiers[i]
#         parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
        parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rb
        clf = GridSearchCV(estimator = clf, param_grid = parameters)

    clf.fit(X_train, y_train)
    clf_ypred = clf.predict(X_test)
    f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
    accuracy_clf = accuracy_score(y_test, clf_ypred)
    print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
    print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*10

    if clf_name == "Support Vector Machine":
        print (clf.best_params_)

```

```
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 64.73802031441801
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 64.80324797001875
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 73.68098312217752
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 74.14116177389131
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 72.8948503457062
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 73.64147407870081
{'C': 3, 'degree': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```

### Building a Model - PCA (n=5)

Multinomial Naive Bayes cannot do PCA as the input is negative

2. Bernoulli NB
3. Logistic Regression
4. Support Vector Machine

In [10]:

```python
Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]

reviews = all_data_cleaned
polarity = data['Polarity']
X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,

# vectorizer = CountVectorizer()
# vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

pca = PCA(n_components = 5)
X_train = pca.fit_transform(X_train.toarray())

df_train = pd.DataFrame(X_train)
df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
df_train.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target']
df_train['pca_1'].replace("", np.nan, inplace = True)
df_train['pca_2'].replace("", np.nan, inplace = True)
df_train['pca_3'].replace("", np.nan, inplace = True)
df_train['pca_4'].replace("", np.nan, inplace = True)
df_train['pca_5'].replace("", np.nan, inplace = True)
df_train['target'].replace("", np.nan, inplace = True)
df_train.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target'], inplace
df_train['pca_1'] = df_train['pca_1'].astype(float)
df_train['pca_2'] = df_train['pca_2'].astype(float)
df_train['pca_3'] = df_train['pca_3'].astype(float)
df_train['pca_4'] = df_train['pca_4'].astype(float)
df_train['pca_5'] = df_train['pca_5'].astype(float)

X_test = pca.transform(X_test.toarray())
df_test = pd.DataFrame(X_test)
df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
df_test.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target']
df_test.describe(include='all')
df_test['pca_1'].replace("", np.nan, inplace = True)
df_test['pca_2'].replace("", np.nan, inplace = True)
df_test['pca_3'].replace("", np.nan, inplace = True)
df_test['pca_4'].replace("", np.nan, inplace = True)
df_test['pca_5'].replace("", np.nan, inplace = True)
df_test['target'].replace("", np.nan, inplace = True)
df_test.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target'], inplace
df_test['pca_1'] = df_test['pca_1'].astype(float)
df_test['pca_2'] = df_test['pca_2'].astype(float)
df_test['pca_3'] = df_test['pca_3'].astype(float)
df_test['pca_4'] = df_test['pca_4'].astype(float)
df_test['pca_5'] = df_test['pca_5'].astype(float)

for i in range(len(Classifiers)):
    clf = Classifiers[i]
    clf_name = "Test"

    if i+1 == int(0):
        clf_name = "Multinomial Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(1):
        clf_name = "Bernoulli Naive Bayes"
        clf = Classifiers[i]
    elif i+1 == int(2):
        clf_name = "Logistic Regression"
        clf = Classifiers[i]
    elif i+1 == int(3):
        clf_name = "Support Vector Machine"
        clf = Classifiers[i]
#         parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
        parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rb
        clf = GridSearchCV(estimator = clf, param_grid = parameters)

    clf.fit(X_train, y_train)
    clf_ypred = clf.predict(X_test)
    f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
    accuracy_clf = accuracy_score(y_test, clf_ypred)
    print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
    print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*100

    if clf_name == "Support Vector Machine":
        print (clf.best_params_)

```

```
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 67.16181696784338
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 67.23922548407245
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 73.69220183933922
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 74.14116177389131
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 73.00631126426943
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 73.70393504059962
{'C': 3, 'degree': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```