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, TfidfVec
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")
```

```
Using TensorFlow backend.
```

Import Data File and Cleaning

In [2]:

```
# data_file = "SHOPEE_MAYBELLINE_CLEAN_V2.csv"
data_file = "Lazada_sentiment.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)

```

Building a Model - Count Vector

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

In [3]:

```
### done

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_)
```

```
F1-score of Multinomial Naive Bayes with Count Vector is: 74.38949627852426
Accuracy of Multinomial Naive Bayes with Count Vector is: 74.88524590163934
F1-score of Bernoulli Naive Bayes with Count Vector is: 78.1315113690941
Accuracy of Bernoulli Naive Bayes with Count Vector is: 78.68852459016394
F1-score of Logistic Regression with Count Vector is: 84.4704797397741
Accuracy of Logistic Regression with Count Vector is: 84.78688524590164
F1-score of Support Vector Machine with Count Vector is: 87.01908836686619
Accuracy of Support Vector Machine with Count Vector is: 87.27868852459017
{'C': 3, 'degree': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```

Building a Model - TFIDF (use_idf = False)

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

In [4]:

```
### done

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_)
```

```
F1-score of Multinomial Naive Bayes with TFIDF (use_idf = False) is: 70.62505296771751
Accuracy of Multinomial Naive Bayes with TFIDF (use_idf = False) is: 72.85245901639344
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = False) is: 78.1315113690941
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = False) is: 78.68852459016394
F1-score of Logistic Regression with TFIDF (use_idf = False) is: 79.84358844245423
Accuracy of Logistic Regression with TFIDF (use_idf = False) is: 81.04918032786885
F1-score of Support Vector Machine with TFIDF (use_idf = False) is: 86.38192827841739
Accuracy of Support Vector Machine with TFIDF (use_idf = False) is: 86.49180327868852
{'C': 3, 'degree': 1, 'gamma': 0.1, 'kernel': 'linear'}
```

Building a Model - TFIDF (use_idf = True)

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

In [5]:

```python
### done

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_)
```

```
F1-score of Multinomial Naive Bayes with TFIDF (use_idf = True) is: 73.45205689395802
Accuracy of Multinomial Naive Bayes with TFIDF (use_idf = True) is: 74.88524590163934
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 78.1315113690941
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 78.68852459016394
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 81.81270330259666
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 82.68852459016394
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 86.52612669729935
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 86.62295081967213
{'C': 3, 'degree': 1, 'gamma': 0.1, 'kernel': 'linear'}
```

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 [6]:

```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: 32.86082463112957
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 49.57377049180327
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 40.10413131989504
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 45.50819672131148
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 32.86082463112957
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 49.57377049180327
{'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 [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 = 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: 51.96431731346798
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 54.81967213114755
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 43.7819197964395
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 47.21311475409836
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 49.4904666008311
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 52.91803278688525
{'C': 2, '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 [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 = 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: 51.96431731346798
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 54.81967213114755
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 42.861789750759925
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 47.21311475409836
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 39.289634468424254
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 48.98360655737705
{'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 [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 = 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*10(

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

```

```
F1-score of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 51.87880776394781
Accuracy of Bernoulli Naive Bayes with TFIDF (use_idf = True) is: 55.47540983606557
F1-score of Logistic Regression with TFIDF (use_idf = True) is: 43.094841922096194
Accuracy of Logistic Regression with TFIDF (use_idf = True) is: 47.278688524590166
F1-score of Support Vector Machine with TFIDF (use_idf = True) is: 39.41974191235565
Accuracy of Support Vector Machine with TFIDF (use_idf = True) is: 48.98360655737705
{'C': 2, 'degree': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```