

Import Packages and Libraries

```
In [1]: 1 from __future__ import absolute_import, division, print_function, unicode_literals
2 import tensorflow as tf
3 from tensorflow.keras import layers
4 from keras.preprocessing.text import Tokenizer
5 from keras.preprocessing.sequence import pad_sequences
6 import numpy as np
7 import pandas as pd
8 import re
9 import gensim
10 from gensim import corpora
11 from gensim import similarities
12 from gensim import models
13 from sklearn.pipeline import Pipeline
14 from sklearn.model_selection import train_test_split, GridSearchCV
15 from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer, TfidfVectorizer
16 from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, f1_score
17 from sklearn.linear_model import LogisticRegression
18 from sklearn.naive_bayes import MultinomialNB, BernoulliNB, GaussianNB
19 from sklearn.svm import SVC
20 from sklearn.decomposition import PCA
21 from sklearn.preprocessing import LabelEncoder
22 import nltk
23 from nltk.corpus import stopwords
24 from nltk.stem.porter import PorterStemmer
25 import warnings
26 warnings.filterwarnings("ignore")
27 from datetime import datetime
```

Using TensorFlow backend.

Import Data File and Cleaning

In [3]:

```
1 # data_file = "SHOPEE_MAYBELLINE_CLEAN_V2.csv"
2 # data_file = "Lazada_sentiment.csv"
3 data_file = "Shopee_AllData_Sentiment_v2.csv"
4 data = pd.read_csv(data_file)
5 data.columns = data.columns.str.strip().str.replace(" ", "_")
6 # data.info()
7 # data.head()
8
9 # data.drop(columns=['Brand', 'Category', 'Product_Name', 'Price', 'Reviewer', 'Product_Purch
10 # review_list = data['Review'].tolist()
11 # polarity_list = data['Polarity'].tolist()
12
13 reviews = data['Review']
14 # polarity = data['Polarity']
15 # print (reviews)
16
17 review_docs = []
18 for each_reviews in reviews:
19     temp = each_reviews.split(" ")
20     review_docs.append(temp)
21 # print (review_docs)
22
23 # Make sure all words are in lowercase
24 reviews_lower = [[each_word.lower() for each_word in each_review] for each_review in re
25 # print (reviews_lower)
26
27 # Use regular expressions to keep only allphabetical words
28 reviews_alpha = [[each_word for each_word in each_review if re.search('^[a-z]+$', each_
29 # print (reviews_alpha)
30
31 # Remove stop words
32 stop_list = stopwords.words('english')
33 reviews_stop = [[each_word for each_word in each_review if each_word not in stop_list]
34 # print (reviews_stop)
35
36 # Porter Stemming
37 stemmer = PorterStemmer()
38 reviews_stem = [[stemmer.stem(each_word) for each_word in each_review] for each_review
39 # print (reviews_stem)
40
41 all_data_cleaned = []
42 for each_sentence in reviews_stem:
43     sentence = ""
44     for each_word in each_sentence:
45         sentence += each_word + " "
46     sentence = sentence[0:-1]
47     all_data_cleaned.append(sentence)
48 # print (all_data_cleaned)
49
50 polarity_raw = data['Polarity']
51 polarity_0_and_1 = []
52 for each_polarity in polarity_raw:
53     if int(each_polarity) == int("0"):
54         polarity_0_and_1.append(0.5)
55     if int(each_polarity) == int("-1"):
56         polarity_0_and_1.append(int(0))
57     if int(each_polarity) == int("1"):
58         polarity_0_and_1.append(int(1))
59 # print (polarity)
60
61 # reviews = all_data_cleaned
62 # print (len(reviews))
63
```

```
64 # product_purchase = data['Product_Purchase']
65 # print (product_purchase[0:10])
66
67
68
```

Building a Model - Count Vector

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

In [4]:

```
1  ### done
2  print (datetime.now())
3
4  Classifiers = [MultinomialNB(), BernoulliNB(), LogisticRegression(), SVC()]
5
6  reviews = all_data_cleaned
7  polarity = data['Polarity']
8  X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,
9
10 countVectorizer = CountVectorizer(min_df = 4, max_df=0.85)
11 X_train = countVectorizer.fit_transform(X_train)
12 X_test = countVectorizer.transform(X_test)
13
14 for i in range(len(Classifiers)):
15     clf = Classifiers[i]
16     clf_name = "Test"
17
18     if i == int(0):
19         clf_name = "Multinomial Naive Bayes"
20         clf = Classifiers[i]
21     elif i == int(1):
22         clf_name = "Bernoulli Naive Bayes"
23         clf = Classifiers[i]
24     elif i == int(2):
25         clf_name = "Logistic Regression"
26         clf = Classifiers[i]
27     elif i == int(3):
28         clf_name = "Support Vector Machine"
29         clf = Classifiers[i]
30     #     parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
31     parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rbf']}
32     clf = GridSearchCV(estimator = clf, param_grid = parameters)
33
34     clf.fit(X_train, y_train)
35     clf_ypred = clf.predict(X_test)
36     f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
37     accuracy_clf = accuracy_score(y_test, clf_ypred)
38     print ("F1-score of", clf_name, "with Count Vector is:", f1_clf*100)
39     print ("Accuracy of", clf_name, "with Count Vector is:", accuracy_clf*100)
40
41     if clf_name == "Support Vector Machine":
42         print (clf.best_params_)
43
44 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]: 1  ### done
2  print (datetime.now())
3
4  Classifiers = [MultinomialNB(), BernoulliNB(), LogisticRegression(), SVC()]
5
6  reviews = all_data_cleaned
7  polarity = data['Polarity']
8  X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,
9
10 tf_Vecorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df=0.85)
11 X_train = tf_Vecorizer.fit_transform(X_train)
12 X_test = tf_Vecorizer.transform(X_test)
13
14 for i in range(len(Classifiers)):
15     clf = Classifiers[i]
16     clf_name = "Test"
17
18     if i == int(0):
19         clf_name = "Multinomial Naive Bayes"
20         clf = Classifiers[i]
21     elif i == int(1):
22         clf_name = "Bernoulli Naive Bayes"
23         clf = Classifiers[i]
24     elif i == int(2):
25         clf_name = "Logistic Regression"
26         clf = Classifiers[i]
27     elif i == int(3):
28         clf_name = "Support Vector Machine"
29         clf = Classifiers[i]
30     #     parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001], '
31     parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rbf']}
32     clf = GridSearchCV(estimator = clf, param_grid = parameters)
33
34     clf.fit(X_train, y_train)
35     clf_ypred = clf.predict(X_test)
36     f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
37     accuracy_clf = accuracy_score(y_test, clf_ypred)
38     print ("F1-score of", clf_name, "with TFIDF (use_idf = False) is:", f1_clf*100)
39     print ("Accuracy of", clf_name, "with TFIDF (use_idf = False) is:", accuracy_clf*100)
40
41     if clf_name == "Support Vector Machine":
42         print (clf.best_params_)
43
44 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]:

```
1  ### done
2  print (datetime.now())
3
4  Classifiers = [MultinomialNB(), BernoulliNB(), LogisticRegression(), SVC()]
5
6  reviews = all_data_cleaned
7  polarity = data['Polarity']
8  X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,
9
10 tfidfVectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
11 X_train = tfidfVectorizer.fit_transform(X_train)
12 X_test = tfidfVectorizer.transform(X_test)
13
14 for i in range(len(Classifiers)):
15     clf = Classifiers[i]
16     clf_name = "Test"
17
18     if i == int(0):
19         clf_name = "Multinomial Naive Bayes"
20         clf = Classifiers[i]
21     elif i == int(1):
22         clf_name = "Bernoulli Naive Bayes"
23         clf = Classifiers[i]
24     elif i == int(2):
25         clf_name = "Logistic Regression"
26         clf = Classifiers[i]
27     elif i == int(3):
28         clf_name = "Support Vector Machine"
29         clf = Classifiers[i]
30     #     parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
31     parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rbf']}
32     clf = GridSearchCV(estimator = clf, param_grid = parameters)
33
34     clf.fit(X_train, y_train)
35     clf_ypred = clf.predict(X_test)
36     f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
37     accuracy_clf = accuracy_score(y_test, clf_ypred)
38     print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
39     print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*100)
40
41     if clf_name == "Support Vector Machine":
42         print (clf.best_params_)
43
44 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]: 1 Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]
2
3 reviews = all_data_cleaned
4 polarity = data['Polarity']
5 X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,
6
7 # vectorizer = CountVectorizer()
8 # vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
9 vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
10 X_train = vectorizer.fit_transform(X_train)
11 X_test = vectorizer.transform(X_test)
12
13 pca = PCA(n_components = 2)
14 X_train = pca.fit_transform(X_train.toarray())
15
16 df_train = pd.DataFrame(X_train)
17 df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
18 df_train.columns = ['pca_1', 'pca_2', 'target']
19 df_train['pca_1'].replace("", np.nan, inplace = True)
20 df_train['pca_2'].replace("", np.nan, inplace = True)
21 df_train['target'].replace("", np.nan, inplace = True)
22 df_train.dropna(subset=['pca_1', 'pca_2', 'target'], inplace = True)
23 df_train['pca_1'] = df_train['pca_1'].astype(float)
24 df_train['pca_2'] = df_train['pca_2'].astype(float)
25
26 X_test = pca.transform(X_test.toarray())
27 df_test = pd.DataFrame(X_test)
28 df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
29 df_test.columns = ['pca_1', 'pca_2', 'target']
30 df_test.describe(include='all')
31 df_test['pca_1'].replace("", np.nan, inplace = True)
32 df_test['pca_2'].replace("", np.nan, inplace = True)
33 df_test['target'].replace("", np.nan, inplace = True)
34 df_test.dropna(subset=['pca_1', 'pca_2', 'target'], inplace = True)
35 df_test['pca_1'] = df_test['pca_1'].astype(float)
36 df_test['pca_2'] = df_test['pca_2'].astype(float)
37
38 for i in range(len(Classifiers)):
39     clf = Classifiers[i]
40     clf_name = "Test"
41
42     if i+1 == int(0):
43         clf_name = "Multinomial Naive Bayes"
44         clf = Classifiers[i]
45     elif i+1 == int(1):
46         clf_name = "Bernoulli Naive Bayes"
47         clf = Classifiers[i]
48     elif i+1 == int(2):
49         clf_name = "Logistic Regression"
50         clf = Classifiers[i]
51     elif i+1 == int(3):
52         clf_name = "Support Vector Machine"
53         clf = Classifiers[i]
54 #     parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
55     parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rbf']}
56     clf = GridSearchCV(estimator = clf, param_grid = parameters)
57
58     clf.fit(X_train, y_train)
59     clf_ypred = clf.predict(X_test)
60     f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
61     accuracy_clf = accuracy_score(y_test, clf_ypred)
62     print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
63     print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*100)

```

```
64
65     if clf_name == "Support Vector Machine":
66         print (clf.best_params_)
67
```

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]: 1 Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]
2
3 reviews = all_data_cleaned
4 polarity = data['Polarity']
5 X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,
6
7 # vectorizer = CountVectorizer()
8 # vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
9 vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
10 X_train = vectorizer.fit_transform(X_train)
11 X_test = vectorizer.transform(X_test)
12
13 pca = PCA(n_components = 3)
14 X_train = pca.fit_transform(X_train.toarray())
15
16 df_train = pd.DataFrame(X_train)
17 df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
18 df_train.columns = ['pca_1', 'pca_2', 'pca_3', 'target']
19 df_train['pca_1'].replace("", np.nan, inplace = True)
20 df_train['pca_2'].replace("", np.nan, inplace = True)
21 df_train['pca_3'].replace("", np.nan, inplace = True)
22 df_train['target'].replace("", np.nan, inplace = True)
23 df_train.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'target'], inplace = True)
24 df_train['pca_1'] = df_train['pca_1'].astype(float)
25 df_train['pca_2'] = df_train['pca_2'].astype(float)
26 df_train['pca_3'] = df_train['pca_3'].astype(float)
27
28 X_test = pca.transform(X_test.toarray())
29 df_test = pd.DataFrame(X_test)
30 df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
31 df_test.columns = ['pca_1', 'pca_2', 'pca_3', 'target']
32 df_test.describe(include='all')
33 df_test['pca_1'].replace("", np.nan, inplace = True)
34 df_test['pca_2'].replace("", np.nan, inplace = True)
35 df_test['pca_3'].replace("", np.nan, inplace = True)
36 df_test['target'].replace("", np.nan, inplace = True)
37 df_test.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'target'], inplace = True)
38 df_test['pca_1'] = df_test['pca_1'].astype(float)
39 df_test['pca_2'] = df_test['pca_2'].astype(float)
40 df_test['pca_3'] = df_test['pca_3'].astype(float)
41
42 for i in range(len(Classifiers)):
43     clf = Classifiers[i]
44     clf_name = "Test"
45
46     if i+1 == int(0):
47         clf_name = "Multinomial Naive Bayes"
48         clf = Classifiers[i]
49     elif i+1 == int(1):
50         clf_name = "Bernoulli Naive Bayes"
51         clf = Classifiers[i]
52     elif i+1 == int(2):
53         clf_name = "Logistic Regression"
54         clf = Classifiers[i]
55     elif i+1 == int(3):
56         clf_name = "Support Vector Machine"
57         clf = Classifiers[i]
58 #     parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
59     parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rbf']}
60     clf = GridSearchCV(estimator = clf, param_grid = parameters)
61
62     clf.fit(X_train, y_train)
63     clf_ypred = clf.predict(X_test)

```

```

64     f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
65     accuracy_clf = accuracy_score(y_test, clf_ypred)
66     print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
67     print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*100)
68
69     if clf_name == "Support Vector Machine":
70         print (clf.best_params_)
71

```

```

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]: 1 Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]
2
3 reviews = all_data_cleaned
4 polarity = data['Polarity']
5 X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,
6
7 # vectorizer = CountVectorizer()
8 # vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
9 vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
10 X_train = vectorizer.fit_transform(X_train)
11 X_test = vectorizer.transform(X_test)
12
13 pca = PCA(n_components = 4)
14 X_train = pca.fit_transform(X_train.toarray())
15
16 df_train = pd.DataFrame(X_train)
17 df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
18 df_train.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target']
19 df_train['pca_1'].replace("", np.nan, inplace = True)
20 df_train['pca_2'].replace("", np.nan, inplace = True)
21 df_train['pca_3'].replace("", np.nan, inplace = True)
22 df_train['pca_4'].replace("", np.nan, inplace = True)
23 df_train['target'].replace("", np.nan, inplace = True)
24 df_train.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target'], inplace = True)
25 df_train['pca_1'] = df_train['pca_1'].astype(float)
26 df_train['pca_2'] = df_train['pca_2'].astype(float)
27 df_train['pca_3'] = df_train['pca_3'].astype(float)
28 df_train['pca_4'] = df_train['pca_4'].astype(float)
29
30 X_test = pca.transform(X_test.toarray())
31 df_test = pd.DataFrame(X_test)
32 df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
33 df_test.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target']
34 df_test.describe(include='all')
35 df_test['pca_1'].replace("", np.nan, inplace = True)
36 df_test['pca_2'].replace("", np.nan, inplace = True)
37 df_test['pca_3'].replace("", np.nan, inplace = True)
38 df_test['pca_4'].replace("", np.nan, inplace = True)
39 df_test['target'].replace("", np.nan, inplace = True)
40 df_test.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'target'], inplace = True)
41 df_test['pca_1'] = df_test['pca_1'].astype(float)
42 df_test['pca_2'] = df_test['pca_2'].astype(float)
43 df_test['pca_3'] = df_test['pca_3'].astype(float)
44 df_test['pca_4'] = df_test['pca_4'].astype(float)
45
46 for i in range(len(Classifiers)):
47     clf = Classifiers[i]
48     clf_name = "Test"
49
50     if i+1 == int(0):
51         clf_name = "Multinomial Naive Bayes"
52         clf = Classifiers[i]
53     elif i+1 == int(1):
54         clf_name = "Bernoulli Naive Bayes"
55         clf = Classifiers[i]
56     elif i+1 == int(2):
57         clf_name = "Logistic Regression"
58         clf = Classifiers[i]
59     elif i+1 == int(3):
60         clf_name = "Support Vector Machine"
61         clf = Classifiers[i]
62 #     parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
63     parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rbf']}

```

```

64         clf = GridSearchCV(estimator = clf, param_grid = parameters)
65
66     clf.fit(X_train, y_train)
67     clf_ypred = clf.predict(X_test)
68     f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
69     accuracy_clf = accuracy_score(y_test, clf_ypred)
70     print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
71     print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*100)
72
73     if clf_name == "Support Vector Machine":
74         print (clf.best_params_)
75

```

```

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]:

```
1 Classifiers = [BernoulliNB(), LogisticRegression(), SVC()]
2
3 reviews = all_data_cleaned
4 polarity = data['Polarity']
5 X_train, X_test, y_train, y_test = train_test_split(reviews, polarity, test_size=0.25,
6
7 # vectorizer = CountVectorizer()
8 # vectorizer = TfidfVectorizer(use_idf = False, min_df = 4, max_df = 0.85)
9 vectorizer = TfidfVectorizer(use_idf = True, min_df = 4, max_df=0.85)
10 X_train = vectorizer.fit_transform(X_train)
11 X_test = vectorizer.transform(X_test)
12
13 pca = PCA(n_components = 5)
14 X_train = pca.fit_transform(X_train.toarray())
15
16 df_train = pd.DataFrame(X_train)
17 df_train = pd.concat([df_train, y_train], axis = 1, ignore_index = True)
18 df_train.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target']
19 df_train['pca_1'].replace("", np.nan, inplace = True)
20 df_train['pca_2'].replace("", np.nan, inplace = True)
21 df_train['pca_3'].replace("", np.nan, inplace = True)
22 df_train['pca_4'].replace("", np.nan, inplace = True)
23 df_train['pca_5'].replace("", np.nan, inplace = True)
24 df_train['target'].replace("", np.nan, inplace = True)
25 df_train.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target'], inplace
26 df_train['pca_1'] = df_train['pca_1'].astype(float)
27 df_train['pca_2'] = df_train['pca_2'].astype(float)
28 df_train['pca_3'] = df_train['pca_3'].astype(float)
29 df_train['pca_4'] = df_train['pca_4'].astype(float)
30 df_train['pca_5'] = df_train['pca_5'].astype(float)
31
32 X_test = pca.transform(X_test.toarray())
33 df_test = pd.DataFrame(X_test)
34 df_test = pd.concat([df_test, y_test], axis = 1, ignore_index = True)
35 df_test.columns = ['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target']
36 df_test.describe(include='all')
37 df_test['pca_1'].replace("", np.nan, inplace = True)
38 df_test['pca_2'].replace("", np.nan, inplace = True)
39 df_test['pca_3'].replace("", np.nan, inplace = True)
40 df_test['pca_4'].replace("", np.nan, inplace = True)
41 df_test['pca_5'].replace("", np.nan, inplace = True)
42 df_test['target'].replace("", np.nan, inplace = True)
43 df_test.dropna(subset=['pca_1', 'pca_2', 'pca_3', 'pca_4', 'pca_5', 'target'], inplace
44 df_test['pca_1'] = df_test['pca_1'].astype(float)
45 df_test['pca_2'] = df_test['pca_2'].astype(float)
46 df_test['pca_3'] = df_test['pca_3'].astype(float)
47 df_test['pca_4'] = df_test['pca_4'].astype(float)
48 df_test['pca_5'] = df_test['pca_5'].astype(float)
49
50 for i in range(len(Classifiers)):
51     clf = Classifiers[i]
52     clf_name = "Test"
53
54     if i+1 == int(0):
55         clf_name = "Multinomial Naive Bayes"
56         clf = Classifiers[i]
57     elif i+1 == int(1):
58         clf_name = "Bernoulli Naive Bayes"
59         clf = Classifiers[i]
60     elif i+1 == int(2):
61         clf_name = "Logistic Regression"
62         clf = Classifiers[i]
63     elif i+1 == int(3):
```

```

64     clf_name = "Support Vector Machine"
65     clf = Classifiers[i]
66     #     parameters = {'C':[1,2,3,4,5,6,7,8,14], 'gamma':[0.1, 0.01, 0.001, 0.0001],
67     parameters = {'C':[1,2,3], 'gamma':[0.1, 0.01], 'kernel':['linear', 'poly', 'rbf']}
68     clf = GridSearchCV(estimator = clf, param_grid = parameters)
69
70     clf.fit(X_train, y_train)
71     clf_ypred = clf.predict(X_test)
72     f1_clf = f1_score(y_test, clf_ypred, average = 'weighted')
73     accuracy_clf = accuracy_score(y_test, clf_ypred)
74     print ("F1-score of", clf_name, "with TFIDF (use_idf = True) is:", f1_clf*100)
75     print ("Accuracy of", clf_name, "with TFIDF (use_idf = True) is:", accuracy_clf*100)
76
77     if clf_name == "Support Vector Machine":
78         print (clf.best_params_)
79

```

```

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'}

```