SMS Spam Filter Using Multinomial and Multivariate Naive Bayes Model

1. Importing and Preprocessing Data

```
import pandas as pd
import numpy as np
from google.colab import files
from sklearn.model_selection  import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn import metrics
import seaborn as sns
import time
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import cross_validate as cvd
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score
```

```
uploaded = files.upload()
```

Choose Files SMSSpamCollection

- **SMSSpamCollection**(n/a) - 477907 bytes, last modified: 11/8/2022 - 100% done

```
Saving SMSSpamCollection to SMSSpamCollection (1)
```

```
# reading the training data
df = pd.read_table('SMSSpamCollection', header=None, names=['Class', 'sms'])
df.head()
```

	Class	sms
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
# number of SMSes / documents
len(df)
```

```
5572
```

```
# counting spam and ham instances
```

```
ham_spam = df.Class.value_counts()
ham_spam
```

```
ham     4825
spam     747
Name: Class, dtype: int64
```

```
print("spam rate is about {0}%".format(
    round((ham_spam[1]/float(ham_spam[0]+ham_spam[1]))*100), 2))
```

```
spam rate is about 13%
```

```
# mapping labels to 0 and 1
df['label'] = df.Class.map({'ham':0, 'spam':1})
```

```
df.head()
```

	Class	sms	label
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

```
# we can now drop the column 'Class'
df = df.drop('Class', axis=1)
df.head()
```

	sms	label
0	Go until jurong point, crazy.. Available only ...	0
1	Ok lar... Joking wif u oni...	0
2	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	U dun say so early hor... U c already then say...	0
4	Nah I don't think he goes to usf, he lives aro...	0

```
# convert to X and y
X = df.sms
y = df.label
print(X.shape)
print(y.shape)
```

```
(5572,)
(5572,)
```

```
# splitting into test and train
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=28)
```

```
X_train.head()
```

```
4387    , im .. On the snowboarding trip. I was wonder...
3491    Huh but i got lesson at 4 lei n i was thinkin ...
3216             I want snow. It's just freezing and windy.
3109    Hello hun how ru? Its here by the way. Im good...
1811    Now, whats your house # again ? And do you hav...
Name: sms, dtype: object
```

```
y_train.head()
```

```
4387    0
3491    0
3216    0
3109    0
1811    0
Name: label, dtype: int64
```

```
# vectorizing the sentences; removing stop words
vect = CountVectorizer(stop_words='english')
```

```
vect.fit(X_train)
```

```
CountVectorizer(stop_words='english')
```

```
# printing the vocabulary
vect.vocabulary_
```

```
 'sonyericsson': 5965,
 'bluetooth': 1352,
 'double': 2323,
 'mobileupd8': 4318,
 '08000839402': 54,
 'call2optout': 1539,
 'f4q': 2620,
 'let': 3874,
 'kanji': 3690,
 'eat': 2411,
 'heavy': 3230,
 'jus': 3671,
 'telling': 6389,
 'leaving': 3852,
 'shanghai': 5731,
 '21st': 358,
 'instead': 3514,
 'haf': 3140,
 'cya': 2051,
 'nope': 4566,
 'fri': 2868,
 'mys': 4427,
 'sis': 5845,
 'paper': 4785,
 'morn': 4352
```

```
      'morn': 4332,
      'ew': 2566,
      'job': 3629,
      'registered': 5352,
      'sinco': 5835,
      'payee': 4831,
      'log': 3963,
      'icicibank': 3413,
      'urn': 6795,
      'confirm': 1879,
      'beware': 1282,
      'frauds': 2847,
      'share': 5735,
      'disclose': 2246,
      'maybe': 4181,
      'leave': 3850,
      'credit': 1984,
      'card': 1578,
      'lar': 3808,
      'testing': 6413,
      'gd': 2957,
      'thk': 6473,
      'bathing': 1200,
      'dun': 2384,
      'disturb': 2261,
      'liao': 3881,
      'cleaning': 1761,
      'dude': 2379,
      'saw': 5599,
      'parked': 4800,
      'sunroof': 6244,
      'popped': 5012,
      'sux': 6276,
      'suffering': 6217,
      'fever': 2702,
```

```
# vocab size
len(vect.vocabulary_.keys())
```

```
     7293
```

```
# transforming the train and test datasets
X_train_transformed = vect.transform(X_train)
X_test_transformed = vect.transform(X_test)
```

```
# note that the type is transformed (sparse) matrix
print(type(X_train_transformed))
print(X_train_transformed)
```

```
     <class 'scipy.sparse.csr.csr_matrix'>
       (0, 830)      1
       (0, 1247)     1
       (0, 1688)     1
       (0, 3089)     1
       (0, 3441)     1
       (0, 3736)     1
       (0, 4203)     1
```

```
  (0, 4942)	1
  (0, 5933)	1
  (0, 6636)	1
  (0, 7134)	1
  (1, 2398)	1
  (1, 3025)	1
  (1, 3051)	1
  (1, 3372)	1
  (1, 3709)	1
  (1, 3862)	1
  (1, 3872)	1
  (1, 4801)	1
  (1, 5613)	1
  (1, 6466)	1
  (1, 6595)	1
  (1, 6824)	1
  (2, 2860)	1
  (2, 3672)	1
  :	:
  (4174, 5408)	1
  (4174, 5681)	1
  (4174, 6092)	1
  (4174, 7246)	1
  (4175, 1277)	1
  (4175, 2086)	1
  (4175, 3035)	1
  (4175, 3263)	1
  (4175, 3318)	1
  (4175, 4534)	1
  (4176, 1828)	1
  (4176, 4568)	1
  (4177, 1236)	1
  (4177, 2096)	1
  (4177, 2896)	1
  (4177, 3025)	1
  (4177, 4110)	1
  (4177, 4491)	1
  (4177, 6136)	1
  (4178, 3290)	1
  (4178, 4647)	1
  (4178, 6514)	1
  (4178, 6699)	1
  (4178, 6800)	1
  (4178, 7146)	1
```

2) Building and Cross-Validation of the Model for Multinomial NB

```
scoring = {'accuracy' : make_scorer(accuracy_score),
           'precision' : make_scorer(precision_score),
           'recall' : make_scorer(recall_score),
           'f1_score' : make_scorer(f1_score)}
```

2.1) 5-fold cross-validation results in terms of accuracy.

```
# training the NB model and making predictions
start = time.time()
mnb = MultinomialNB()
#cross validtion
score1 = cross_val_score(mnb, X_train_transformed,y_train, cv=5, scoring='accuracy')
print("Average Cross Validation Accuracy for 5-Folds using Multinomial Naive Bayes:-",np.mean(score1))
```

```
Average Cross Validation Accuracy for 5-Folds using Multinomial Naive Bayes:- 0.9775079506059651
```

2.2) 10-fold cross-validation results in terms of precision, recall,and F-score

```
score1 = cvd(mnb, X_train_transformed,y_train, cv=10, scoring=scoring)
print("Average Cross Validation precision for 10-Folds using Multinomial Naive Bayes:-",score1['test_precision'].mean())
print("Average Cross Validation recall for 10-Folds using Multinomial Naive Bayes:-",score1['test_recall'].mean())
print("Average Cross Validation F1 score for 10-Folds using Multinomial Naive Bayes:-",score1['test_f1_score'].mean())
end =time.time()
print("Time taken:-",end-start)
```

```
Average Cross Validation precision for 10-Folds using Multinomial Naive Bayes:- 0.8938246352802433
Average Cross Validation recall for 10-Folds using Multinomial Naive Bayes:- 0.9599213551119176
Average Cross Validation F1 score for 10-Folds using Multinomial Naive Bayes:- 0.9253089713967491
Time taken:- 0.1282963752746582
```

2.3) Training Multinomial NB Model

```
# fit
mnb.fit(X_train_transformed,y_train)

# predict class
y_pred_class1 = mnb.predict(X_test_transformed)

# predict probabilities
y_pred_proba1 = mnb.predict_proba(X_test_transformed)

#time taken for training and cross validation
end =time.time()
print("Time taken:-",end-start)
```

```
Time taken:- 0.15954852104187012
```

```
y_pred_proba1
```

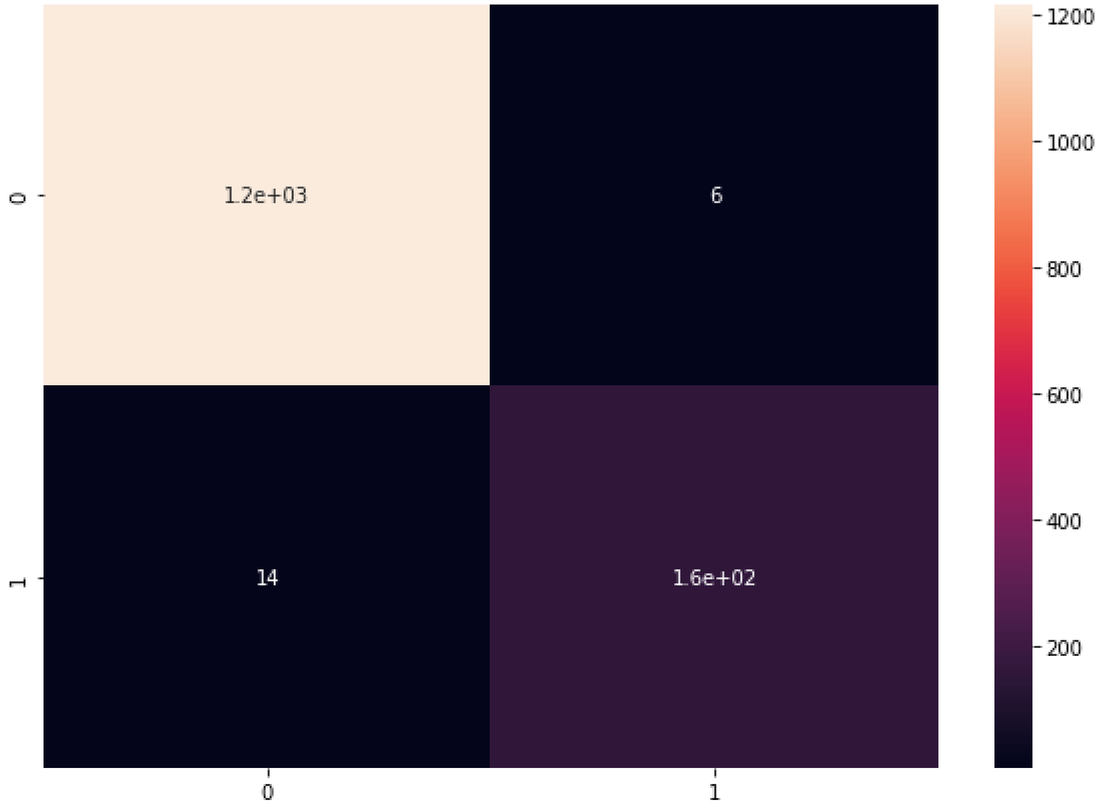
```
array([[9.99999999e-01, 1.04542272e-09],
       [9.99968215e-01, 3.17847344e-05],
       [9.88816562e-01, 1.11834379e-02],
       ...,
       [9.99998916e-01, 1.08355798e-06],
       [3.28345400e-01, 6.71654600e-01],
       [9.99893544e-01, 1.06456286e-04]])
```

2.4) Model Evaluation

```
# printing the overall accuracy
metrics.accuracy_score(y_test, y_pred_class1)
```

```
0.9856424982053122
```

```
# confusion matrix
cm=metrics.confusion_matrix(y_test, y_pred_class1)
plt.figure(figsize = (10,7))
sns.heatmap(cm, annot=True)
TN = cm[0, 0]
FP = cm[0, 1]
FN = cm[1, 0]
TP = cm[1, 1]
```



```
sensitivity = TP / float(FN + TP)
print("sensitivity",sensitivity)
specificity = TN / float(TN + FP)
print("specificity",specificity)
precision = TP / float(TP + FP)
print("precision",precision)
print(metrics.precision_score(y_test, y_pred_class1))
print("precision",precision)
print("PRECISION SCORE :",metrics.precision_score(y_test, y_pred_class1))
print("RECALL SCORE :", metrics.recall_score(y_test, y_pred_class1))
print("F1 SCORE :",metrics.f1_score(y_test, y_pred_class1))
```

```
sensitivity 0.9190751445086706
specificity 0.9950819672131147
precision 0.9636363636363636
0.9636363636363636
```

```
precision 0.9636363636363636
PRECISION SCORE : 0.9636363636363636
RECALL SCORE : 0.9190751445086706
F1 SCORE : 0.9408284023668639
```

```
# creating an ROC curve
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred_proba1[:,1])
roc_auc = auc(false_positive_rate, true_positive_rate)
```

```
# area under the curve
print (roc_auc)
```

```
0.9877996778167346
```

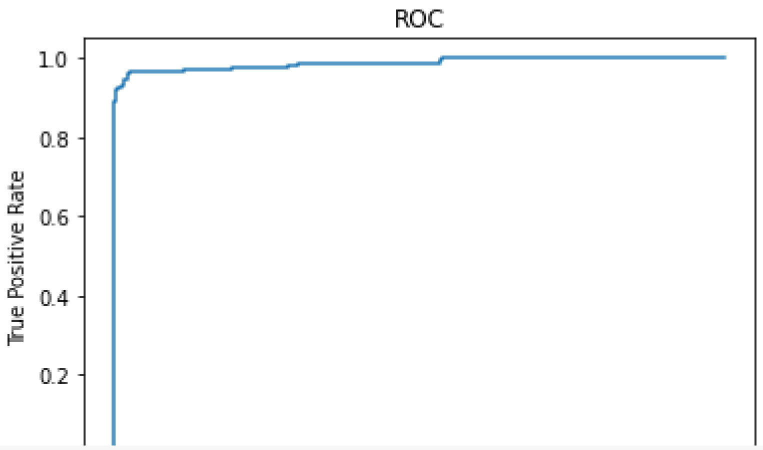
```
# matrix of thresholds, tpr, fpr
pd.DataFrame({'Threshold': thresholds,
              'TPR': true_positive_rate,
              'FPR':false_positive_rate
             })
```

	Threshold	TPR	FPR
0	2.000000e+00	0.000000	0.000000
1	1.000000e+00	0.335260	0.000000
2	1.000000e+00	0.341040	0.000000
3	1.000000e+00	0.352601	0.000000
4	1.000000e+00	0.381503	0.000000
...	...	...	...
116	7.657420e-09	1.000000	0.873770
117	7.391974e-09	1.000000	0.875410
118	1.439132e-09	1.000000	0.897541
119	1.353629e-09	1.000000	0.899180
120	4.254529e-41	1.000000	1.000000

121 rows × 3 columns

```
# plotting the ROC curve
%matplotlib inline
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC')
plt.plot(false_positive_rate, true_positive_rate)
```

```
[<matplotlib.lines.Line2D at 0x7f5e3c103310>]
```



```
input1 = ['Submit AI Assignment']
input1_transform = vect.transform(input1)
print(mnb.predict(input1_transform))
```

```
[0]
```

```
#input2 = ['Will pick you at 7pm']
input2 =['Free entry in 2 a wkly comp to win cricket wrld cup final']
input2_transform = vect.transform(input2)
print(mnb.predict(input2_transform))
```

```
[1]
```

3) Building and Cross-Validation of the Model for Multivariate NB

3.1) 5-fold cross-validation results in terms of accuracy.

```
start = time.time()
mvb=BernoulliNB()
#cross validtion
score2 = cross_val_score(mvb, X_train_transformed,y_train, cv=5, scoring='accuracy')
print("Average Cross Validation Accuracy for 5-Folds using Multivariate Naive Bayes:-",np.mean(score2))
```

```
Average Cross Validation Accuracy for 5-Folds using Multivariate Naive Bayes:- 0.9734395324184166
```

3.2) 10-fold cross-validation results in terms of precision, recall,and F-score

```
score2 = cvd(mvb, X_train_transformed,y_train, cv=10, scoring=scoring)
print("Average Cross Validation precision for 10-Folds using Multivariate Naive Bayes:-",score2['test_precision'].mean())
print("Average Cross Validation recall for 10-Folds using Multivariate Naive Bayes:-",score2['test_recall'].mean())
print("Average Cross Validation F1 score for 10-Folds using Multivariate Naive Bayes:-",score2['test_f1_score'].mean())
```

```
Average Cross Validation precision for 10-Folds using Multivariate Naive Bayes:- 0.9733145572019092
Average Cross Validation recall for 10-Folds using Multivariate Naive Bayes:- 0.857047791893527
Average Cross Validation F1 score for 10-Folds using Multivariate Naive Bayes:- 0.9104419163408396
```

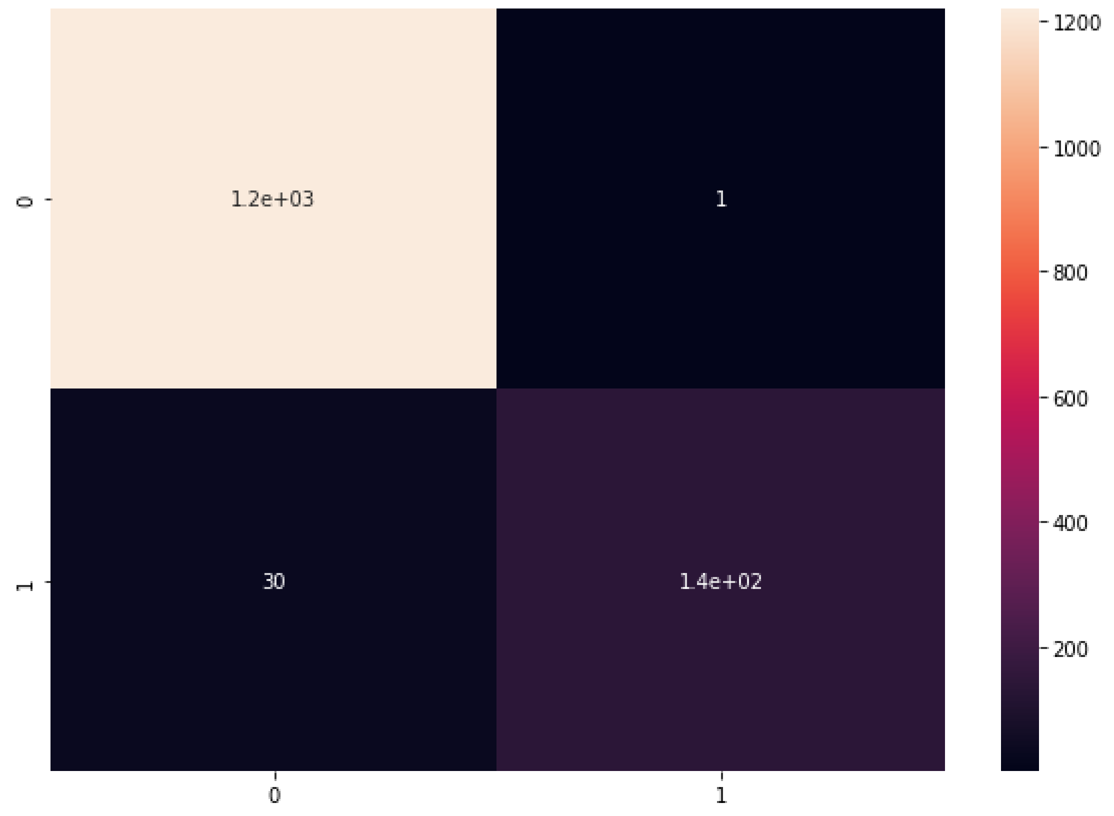
3.3) Training Multivariate NB Model

```
# fit
mvb.fit(X_train_transformed,y_train)

# predict class
y_pred_class2 = mvb.predict(X_test_transformed)

# predict probabilities
y_pred_proba2 = mvb.predict_proba(X_test_transformed)

#time taken for training and cross validation
end =time.time()
print("Time taken:-",end-start)
```

```
Time taken:- 0.17893671989440918
```

3.4) Model Evaluation

```
# confusion matrix
cm=metrics.confusion_matrix(y_test, y_pred_class2)
plt.figure(figsize = (10,7))
sns.heatmap(cm, annot=True)
TN = cm[0, 0]
FP = cm[0, 1]
FN = cm[1, 0]
TP = cm[1, 1]
```



```
sensitivity = TP / float(FN + TP)
print("sensitivity",sensitivity)
specificity = TN / float(TN + FP)
print("specificity",specificity)
precision = TP / float(TP + FP)
print("precision",precision)
print(metrics.precision_score(y_test, y_pred_class1))
print("precision",precision)
print("PRECISION SCORE :",metrics.precision_score(y_test, y_pred_class1))
print("RECALL SCORE :", metrics.recall_score(y_test, y_pred_class1))
print("F1 SCORE :",metrics.f1_score(y_test, y_pred_class1))
```

```
sensitivity 0.8265895953757225
specificity 0.9991803278688525
precision 0.9930555555555556
0.9636363636363636
precision 0.9930555555555556
PRECISION SCORE : 0.9636363636363636
RECALL SCORE : 0.9190751445086706
F1 SCORE : 0.9408284023668639
```

```
# creating an ROC curve
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred_proba2[:,1])
roc_auc = auc(false_positive_rate, true_positive_rate)
```

```
# area under the curve
print (roc_auc)
```

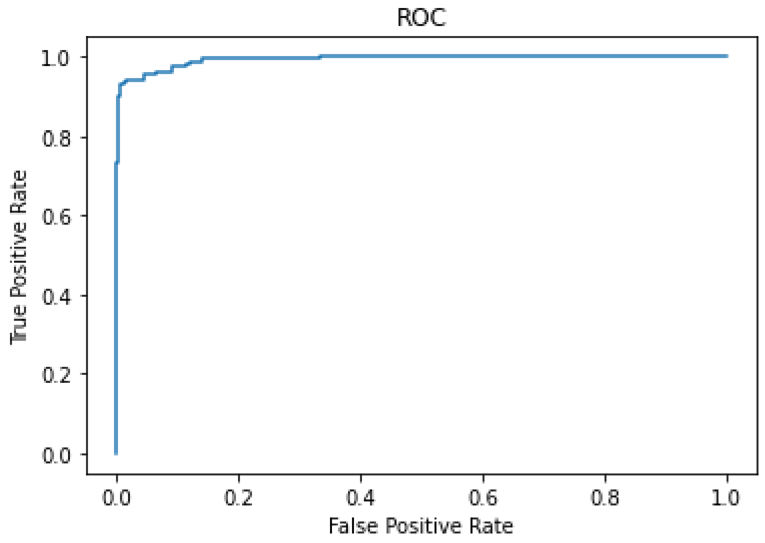
```
0.9928740642471334
```

```
# matrix of thresholds, tpr, fpr
pd.DataFrame({'Threshold': thresholds,
              'TPR': true_positive_rate,
              'FPR':false_positive_rate
             })
```

	Threshold	TPR	FPR
0	2.000000e+00	0.000000	0.000000
1	1.000000e+00	0.277457	0.000000

```
# plotting the ROC curve
%matplotlib inline
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC')
plt.plot(false_positive_rate, true_positive_rate)
```

```
[<matplotlib.lines.Line2D at 0x7f5e3bf1d450>]
```



```
input1 = ['Submit AI Assignment']
input1_transform = vect.transform(input1)
print(mvb.predict(input1_transform))
```

```
[0]
```

```
#input2 = ['Will pick you at 7pm']
input2 =['Free entry in 2 a wkly comp to win cricket Cup final']
input2_transform = vect.transform(input2)
print(mvb.predict(input2_transform))
```

```
[1]
```

✓ 0s completed at 3:16 PM ● ×