Text Representation - Bag Of Words (BOW)

AIM:To use Bag of words encoding technique and train using several Machine learning Models.

Description: 1. The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. Machine Learning Models used: 1. Naive Bayes: Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. 2. Randomforest classifier:A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
3. SVM :The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

4. KNN : K-Nearest Neighbors Algorithm. The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

```
import pandas as pd
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Double-click (or enter) to edit

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/NLP/revpre.csv')
```

```
df
```

	Unnamed: 0.1	Unnamed: 0	Review	Rating	Recommended IND	Positive Feedback Count
0	0	0	'absolutely wonderful silky sexy comfortable '	4	1	0
1	1	1	'love dress sooo pretty happened find store im...	5	1	4
2	2	2	'high hopes dress really wanted work initially...	3	0	0
3	3	3	'love love love jumpsuit fun flirty fabulous e...	5	1	0
4	4	4	'this shirt flattering due adjustable front ti...	5	1	6
...	...	...	...	...	...	...
23481	23481	23481	'happy snag dress great price easy slip flatte...	5	1	0
23482	23482	23482	'it reminds maternity clothes soft stretchy sh...	3	1	0
23483	23483	23483	'this fit well top see never would worked im g...	3	0	1
23484	23484	23484	'bought dress wedding summer cute unfortunatel...	3	1	2
23485	23485	23485	'this dress lovely platinum feminine fits perf...	5	1	22

```
df.dropna()
```

	Unnamed: 0.1	Unnamed: 0	Review	Rating	Recommended IND	Positive Feedback Count
0	0	0	'absolutely wonderful silky sexy comfortable '	4	1	0
1	1	1	'love dress sooo pretty happened find store im...	5	1	4
2	2	2	'high hopes dress really wanted work initially...	3	0	0
3	3	3	'love love love jumpsuit fun flirty fabulous e...	5	1	0

```
df.dropna(how='all')
```

	Unnamed: 0.1	Unnamed: 0	Review	Rating	Recommended IND	Positive Feedback Count
0	0	0	'absolutely wonderful silky sexy comfortable '	4	1	0
1	1	1	'love dress sooo pretty happened find store im...	5	1	4
2	2	2	'high hopes dress really wanted work initially...	3	0	0
3	3	3	'love love love jumpsuit fun flirty fabulous e...	5	1	0
4	4	4	'this shirt flattering due adjustable front ti...	5	1	6
...	...	...	...	...	...	...
23481	23481	23481	'happy snag dress great price easy slip flatte...	5	1	0
23482	23482	23482	'it reminds maternity clothes soft stretchy sh...	3	1	0
23483	23483	23483	'this fit well top see never would worked im g...	3	0	1
23484	23484	23484	'bought dress wedding summer cute unfortunatel...	3	1	2
23485	23485	23485	'this dress lovely platinum feminine fits perf...	5	1	22

```
df.shape
```

```
(23486, 6)
```

```
df.head()
```

	Unnamed: 0.1	Unnamed: 0	Review	Rating	Recommended IND	Positive Feedback Count
0	0	0	'absolutely wonderful silky sexy comfortable '	4	1	0
1	1	1	'love dress sooo pretty happened find store im...	5	1	4
2	2	2	'high hopes dress really wanted work initially...	3	0	0
3	3	3	'love love love jumpsuit fun flirty fabulous e...	5	1	0
4	4	4	'this shirt flattering due adjustable front ti...	5	1	6

```
df.columns
```

```
Index(['Unnamed: 0.1', 'Unnamed: 0', 'Review', 'Rating', 'Recommended IND',
       'Positive Feedback Count'],
      dtype='object')
```

Train test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.Review, df.Rating, test_size=0.2)
```

```
X_train.shape
```

```
(18788,)
```

```
X_test.shape
```

```
(4698,)
```

```
type(X_train)
```

```
pandas.core.series.Series
```

```
X_train[:4]
```

```
23242    'bought jeans absolutely love new favorites li...
15445    'love style fit great buy legs arent long mode...
16009    'these go work pants 10 pairs disappointed mak...
543      'based reviews decided get regular xs even tho...
Name: Review, dtype: object
```

```
type(y_train)
```

```
pandas.core.series.Series
```

```
y_train[:4]
```

```
23242    5
15445    4
16009    5
543      4
Name: Rating, dtype: int64
```

```
type(X_train.values)
```

```
numpy.ndarray
```

```
val=df[df['Review']==' ']
print(val)
```

```
Empty DataFrame
Columns: [Unnamed: 0.1, Unnamed: 0, Review, Rating, Recommended IND, Positive Feedback Count]
Index: []
```

```
df.isna().any()
```

```
Unnamed: 0.1               False
Unnamed: 0                 False
Review                     False
Rating                     False
Recommended IND            False
Positive Feedback Count    False
dtype: bool
```

```
df.dropna()
```

	Unnamed: 0.1	Unnamed: 0	Review	Rating	Recommended IND	Positive Feedback Count
0	0	0	'absolutely wonderful silky sexy comfortable '	4	1	0
1	1	1	'love dress sooo pretty happened find store im...	5	1	4
2	2	2	'high hopes dress really wanted work initially...	3	0	0
3	3	3	'love love love jumpsuit fun flirty fabulous e...	5	1	0
4	4	4	'this shirt flattering due adjustable front ti...	5	1	6
...	...	...	...	...	...	...
23481	23481	23481	'happy snag dress great price easy slip flatte...	5	1	0
23482	23482	23482	'it reminds maternity clothes soft stretchy sh...	3	1	0
23483	23483	23483	'this fit well top see never would worked im g...	3	0	1
23484	23484	23484	'bought dress wedding summer cute unfortunatel...	3	1	2
23485	23485	23485	'this dress lovely platinum feminine fits perf...	5	1	22

```
df.isna().any()
```

```
Unnamed: 0.1               False
Unnamed: 0                 False
Review                     False
Rating                     False
Recommended IND            False
Positive Feedback Count    False
dtype: bool
```

Create bag of words representation using CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
X_train_cv =v.fit_transform(X_train.values.astype(str))
X_train_cv
```

```
<18788x17225 sparse matrix of type '<class 'numpy.int64'>'
        with 494879 stored elements in Compressed Sparse Row format>
```

```
X_train_cv.toarray()[:2][0]
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
X_train_cv.shape
```

```
(18788, 17225)
```

```
dir(v)
```

```
['__annotations__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__setstate__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_char_ngrams',
 '_char_wb_ngrams',
 '_check_feature_names',
 '_check_n_features',
 '_check_stop_words_consistency',
 '_check_vocabulary',
 '_count_vocab',
 '_get_param_names',
 '_get_tags',
 '_limit_features',
 '_more_tags',
 '_parameter_constraints',
 '_repr_html_',
 '_repr_html_inner',
 '_repr_mimebundle_',
 '_sort_features',
 '_stop_words_id',
 '_validate_data',
 '_validate_ngram_range',
 '_validate_params',
 '_validate_vocabulary',
 '_warn_for_unused_params',
 '_white_spaces',
 '_word_ngrams',
 'analyzer',
 'binary',
 'build_analyzer',
 'build_preprocessor',
 'build_tokenizer',
```

```
v.get_feature_names_out()[2678]
```

```
'budge'
```

```
v.vocabulary_
```

```
      'dream': 4943,
      'sale': 12660,
      'excited': 5519,
      'jumper': 8101,
      'huge': 7419,
      '58': 734,
      '130lbs': 166,
      'perhaps': 10924,
      'upper': 16158,
      'body': 2332,
      'tightstretched': 15370,
      'print': 11549,
      '100': 20,
      'less': 8542,
      'current': 4162,
      'price': 11519,
      'needs': 9879,
      'perfectly': 10901,
      'sadly': 12630,
      'looking': 8800,
      'jacket': 7961,
      'line': 8638,
      'finally': 5913,
      'went': 16688,
      'styling': 14499,
      'stiff': 14243,
      'id': 7495,
      'prefer': 11439,
      'softer': 13862,
      'kind': 8187,
      'wonder': 16927,
      'soften': 13857,
      'ran': 11858,
      'surprisingly': 14690,
      'usually': 16199,
      'xsmall': 17081,
      'leave': 8458,
      'room': 12507,
      'comfortably': 3581,
      'bend': 2038,
      'medium': 9239,
      'others': 10441,
      'ill': 7523,
      'wait': 16439,
      'goes': 6673,
      'sal': 12658,
      'booties': 2397,
      '54': 703,
      '12': 109,
      'hits': 7270,
      'concerned': 3687,
      'lifestyle': 8573,
      'photo': 10996,
      'roll': 12485,
      'tab': 14846,
      'button': 2795,
      'low': 8910,
      'therefore': 15132,
      'sleeve': 13573,
```

```
X_train_np = X_train_cv.toarray()
X_train_np[0]
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
np.where(X_train_np[0]!=0)
```

```
(array([  888,  1243,  1333,  2460,  2827,  5457,  5794,  7111,  8009,
         8609,  8884,  9689,  9927, 10629, 10890, 13214, 13625, 16550,
        16570, 16655]),)
```

```
X_train_np[0][3256]
```

```
0
```

Train the naive bayes model

```
missing_labels = np.isnan(y_train)
X_train_cv = X_train_cv[~missing_labels]
y_train = y_train[~missing_labels]
print(X_train_cv.shape[0] == y_train.shape[0])
```

```
True
```

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_np, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
X_test_cv = v.transform(X_test.values.astype('U'))
```

```
from sklearn.metrics import classification_report

y_pred = model.predict(X_test_cv)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.36      0.02      0.05       166
           2       0.32      0.07      0.12       289
           3       0.40      0.39      0.40       581
           4       0.41      0.33      0.37      1007
           5       0.75      0.92      0.83      2655

    accuracy                           0.64      4698
   macro avg       0.45      0.35      0.35      4698
weighted avg       0.59      0.64      0.60      4698
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
from sklearn import metrics
#1. create a pipeline object
clf = Pipeline([
     ('vectorizer', CountVectorizer(ngram_range = (1, 6))),        #using the ngram_range parameter
     ('nb', MultinomialNB())
])

#2. fit with X_train and y_train
clf.fit(X_train, y_train)


#3. get the predictions for X_test and store it in y_pred
y_pred = clf.predict(X_test)


#4. print the classfication report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       166
           2       1.00      0.00      0.01       289
           3       0.47      0.02      0.03       581
           4       0.40      0.01      0.02      1007
           5       0.57      1.00      0.73      2655

    accuracy                           0.57      4698
   macro avg       0.49      0.21      0.16      4698
weighted avg       0.53      0.57      0.42      4698
```

```
y_test
```

```
1188     5
6679     5
20215    4
1543     5
17639    4
        ..
9694     5
13948    5
13300    5
9813     3
```

```
10223    5
Name: Rating, Length: 4698, dtype: int64
```

```
y_pred
```

```
array([5, 5, 5, ..., 5, 5, 5])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

v = TfidfVectorizer()

X_train_t = v.fit_transform(X_train.values)
X_train_t
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_t, y_train)
X_test_t = v.transform(X_test)
from sklearn.metrics import classification_report

y_pred = model.predict(X_test_t)
print("multinomialNB Accuracy:",metrics.accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
multinomialNB Accuracy: 0.5655598126862494
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       166
           2       0.00      0.00      0.00       289
           3       0.14      0.00      0.00       581
           4       0.15      0.00      0.00      1007
           5       0.57      1.00      0.72      2655

    accuracy                           0.57      4698
   macro avg       0.17      0.20      0.15      4698
weighted avg       0.37      0.57      0.41      4698

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

Support vector classifier with linear kernel gave maximum accuracy of 0.69 . Random forest classifier gave 0.68 accuracy. Accuracy with Naive bayes is 0.57 , with SVM poly kernel is 0.59, with SVM rbf kernel is 0.68, with knn is 0.39