EXPERIMENT NO. 6

AIM: Study and Implement Naive Bayes.

Theory:

1. **Naive Bayes Classification:**

- Naive Bayes is a family of simple probabilistic classifiers based on Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to the event.
- This classifier is termed "naive" because it assumes that all features (predictors) are independent of each other given the target class, which is often not true in practice. However, this independence assumption simplifies computation, making Naive Bayes efficient and scalable.
- Naive Bayes classifiers are particularly effective for large datasets, especially when working with high-dimensional data, as is common in text classification tasks like spam detection, sentiment analysis, and document categorization.

2. **Bayes' Theorem:**

- Bayes' theorem is the core formula of Naive Bayes and is mathematically expressed as:

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$$

where:

- P(Y|X): Posterior probability of class Y given predictor X
- P(X|Y): Likelihood of predictor X given class Y
- P(Y): Prior probability of class Y
- P(X): Prior probability of predictor X

Code:

```
# In[1]:
import pandas as pd

# In[2]:
df = pd.read_csv("spam.csv")
df.head()

# In[3]:
```

```
df.groupby('Category').describe()

#In[4]:
df['spam']=df['Category'].apply(lambda x: 1 if x=='spam' else 0)
df.head()
#In[5]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.Message,df.spam)

#In[6]:
from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
X_train_count = v.fit_transform(X_train.values)
X_train_count.toarray()[:2]

#In[7]:
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_count,y_train)

#In[8]:
emails = [
    'Hey mohan, can we get together to watch footbal game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you. Dont miss this reward!'
]
emails_count = v.transform(emails)
model.predict(emails_count)

#In[9]:
X_test_count = v.transform(X_test)
model.score(X_test_count, y_test)
```

Sklearn Pipeline

```
#In[10]:
from sklearn.pipeline import Pipeline
clf = Pipeline([
('vectorizer', CountVectorizer()),
('nb', MultinomialNB())
])

#In[11]:
clf.fit(X_train, y_train)
```

```
#In[12]:
clf.score(X_test,y_test)

#In[13]:
clf.predict(emails)
```

Output Snapshots:

	Category	Message
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...

	Message			
	count	unique	top	freq
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	641	Please call our customer service representativ...	4

	Category	Message	spam
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

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
array([0, 1], dtype=int64)    0.9827709978463748
```

```
Pipeline(memory=None,
     steps=[('vectorizer', CountVectorizer(analyzer='word', binary=False, decode
_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)), ('nb', MultinomialNB(alpha=1.0, class
_prior=None, fit_prior=True))])
```

```
0.9827709978463748
```

```
array([0, 1], dtype=int64)
```

Learning Outcome: