ML life cycle

1.Frame the problem

2.Gathering Data

3.Data Preprocessing

4.EDA

5.Featuring eng and selc

6.model Training

7.Model Deployment

In [1]:
```
import pandas as pd
import numpy as np
import sklearn
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data = pd.read_csv("spam.csv", encoding = "latin-1")
```

In [3]:
```
data.sample(4)
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
1057	ham	Ard 515 like dat. Y?	NaN	NaN	NaN
1827	ham	Hey gorgeous man. My work mobile number is. Ha...	NaN	NaN	NaN
79	ham	Its not the same here. Still looking for a job...	NaN	NaN	NaN
5126	ham	To the wonderful Okors, have a great month. We...	NaN	NaN	NaN

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

In [5]:
```
data.drop(['Unnamed: 2', 'Unnamed: 4', 'Unnamed: 3'], inplace = True, axis = 1)
```

In [6]:
```
data.sample(3)
```

Out[6]:

	v1	v2
1415	ham	Jay is snickering and tells me that x is total...
150	ham	The wine is flowing and i'm i have nevering..
805	ham	I dled 3d its very imp

In [7]:
```
data.rename(columns = {'v1':'class', 'v2':'text'}, inplace = True)
```

In [8]:
```
data.sample(1)
```

Out[8]:

	class	text
282	ham	Ok. I asked for money how far

In [9]:
```
data  = data.reindex(columns = ['text', 'class'])
```

In [10]:
```
data.sample(2)
```

Out[10]:

	text	class
2248	will you like to be spoiled? :)	ham
5370	dating:i have had two of these. Only started a...	spam

In [11]:
```
data.isna().sum()
```

Out[11]:
```
text     0
class    0
dtype: int64
```

In [12]:
```
data = data.drop_duplicates(keep='first')
```

In [13]:
```
data.shape
```

Out[13]:
```
(5169, 2)
```

In [ ]:
```

```

label encoder

In [14]:
```
from sklearn.preprocessing import LabelEncoder
encode = LabelEncoder()
```

In [15]:
```
data['class'] = encode.fit_transform(data['class'])
```

In [16]:
```
encode.classes_
```

Out[16]:
```
array(['ham', 'spam'], dtype=object)
```

In [ ]:
```

```

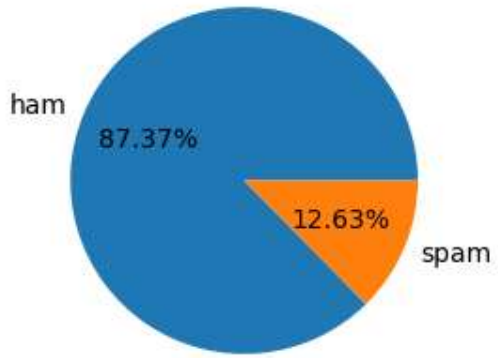
EDA

In [17]:
```
#dataset balance
import matplotlib.pyplot as plt
data['class'].value_counts()
```

Out[17]:
```
class
0    4516
1     653
Name: count, dtype: int64
```

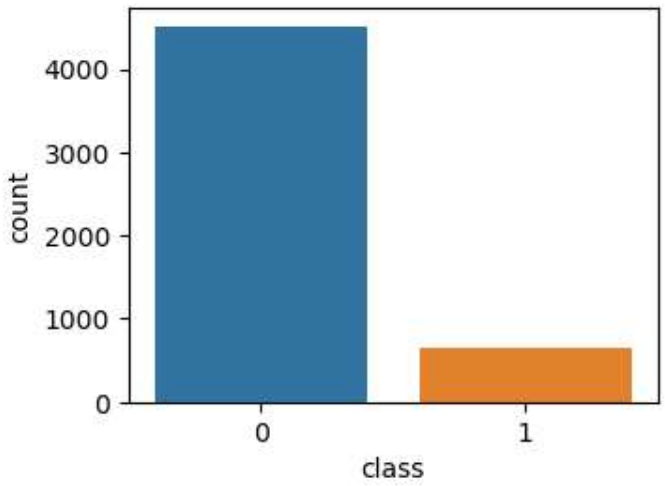
In [18]:
```
# data['class'].value_counts().plot(kind='pie')
```

In [19]:
```
plt.figure(figsize=(6, 3))
plt.pie(data['class'].value_counts(), labels=['ham', 'spam'], autopct='%1.2f%%')
plt.show()
```



In [20]:
```
import seaborn as sns
fig = plt.figure(figsize=(8, 6))
p1 = fig.add_subplot(2, 2, 1)
sns.countplot(data, x='class')
```

Out[20]: <Axes: xlabel='class', ylabel='count'>



In [21]:
```
# sns.histplot(data, x = 'class')
```

text processing

1.lowercase

2.stopwords

3.special chars

4.tokens

5.Stemming

6.vectorization

In [22]:
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize, wordpunct_tokenize
import string
from nltk.stem import PorterStemmer
```

In [23]:
```
def words(text):
    w = word_tokenize(text)
    return len(w)
def sents(text):
    s = sent_tokenize(text)
    return len(s)

data['no_words'] = data['text'].apply(words)
data['no_sents'] = data['text'].apply(sents)
```

In [24]:
```
data.head(2)
```

Out[24]:

	text	class	no_words	no_sents
0	Go until jurong point, crazy.. Available only ...	0	24	2
1	Ok lar... Joking wif u oni...	0	8	2

In [126]:
```
print(data[data["class"]==0]["text"][39])
```

```
Hello! How's you and how did saturday go? I was just texting to see if you'd decided t
o do anything tomo. Not that i'm trying to invite myself or anything!
```

In [25]:
```
accents = "Nátùrâl Lânguáge Pythön Mâchinè Lèárning Dèèp Lèárning"
acc = "Héy Dánc Ládí Amáz Èvèn, fór thís"
```

In [26]:
```
from unidecode import unidecode
print(unidecode(acc))
print(unidecode(accents))
```

```
Hey Danc Ladi Amaz Even, for this
Natural Language Python Machine Learning Deep Learning
```

In [27]:
```
string.punctuation
```

Out[27]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [28]:
```
def convert(text):
    text = unidecode(text.lower())
    stops = stopwords.words('english')
    y = []
    for i in wordpunct_tokenize(text):
        if (i not in stops and i not in string.punctuation):
            y.append(i)

    text = y.copy()
    y.clear()

    for i in text:
        if(i.isalnum()):
            y.append(i)

    text = y.copy()
    y.clear()
    stemmer = PorterStemmer()
    text = [stemmer.stem(word) for word in text]
    return " ".join(text)
```

In [29]:
```
convert("hey dancing lady you are amazing in this evening")
```

Out[29]:
```
'hey danc ladi amaz even'
```

In [30]:
```
data['processed_text'] = data['text'].apply(convert)
```

In [31]:
```
data['text'][0]
```

Out[31]:
```
'Go until jurong point, crazy.. Available only in bugis n great world la e buffet... C
ine there got amore wat...'
```

In [32]:
```
data['processed_text'][0]
```

Out[32]:
```
'go jurong point crazi avail bugi n great world la e buffet cine got amor wat'
```

In [33]:
```
data['no_proc_words'] = data['processed_text'].apply(words)
data['no_proc_sents'] = data['processed_text'].apply(sents)
```

In [34]:
```
data.head(2)
```

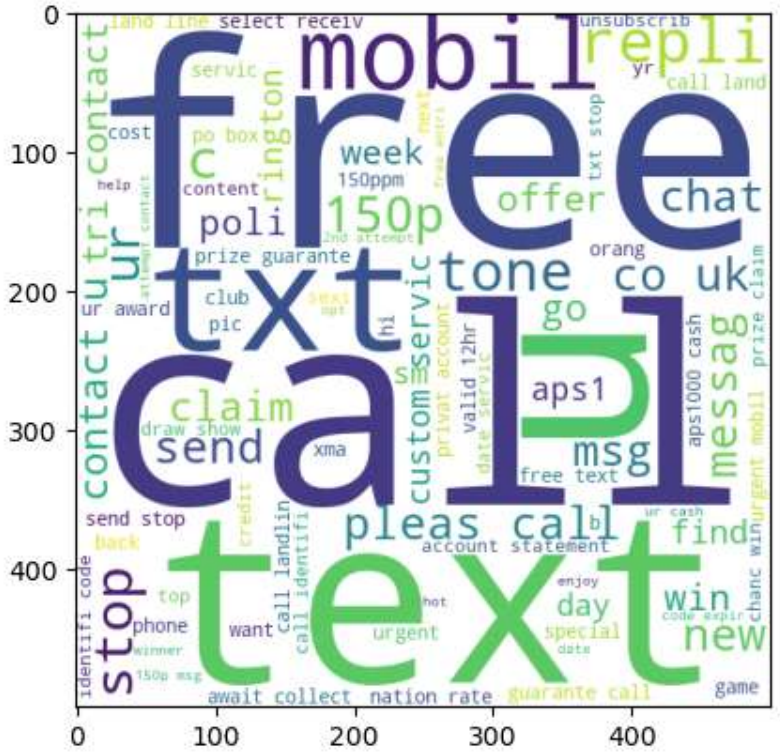
Out[34]:

	text	class	no_words	no_sents	processed_text	no_proc_words	no_proc_sents
0	Go until jurong point, crazy.. Available only ...	0	24	2	go jurong point crazi avail bugi n great world...	16	1
1	Ok lar... Joking wif u oni...	0	8	2	ok lar joke wif u oni	6	1

In [35]:
```
# pip install wordcloud
```

In [36]:
```
#WordCloud
from wordcloud import WordCloud
wc  = WordCloud(width=500, height=500, min_font_size=10, background_color='white')
spam_wc = wc.generate(data[data['class']==1]['processed_text'].str.cat(sep=" "))
```

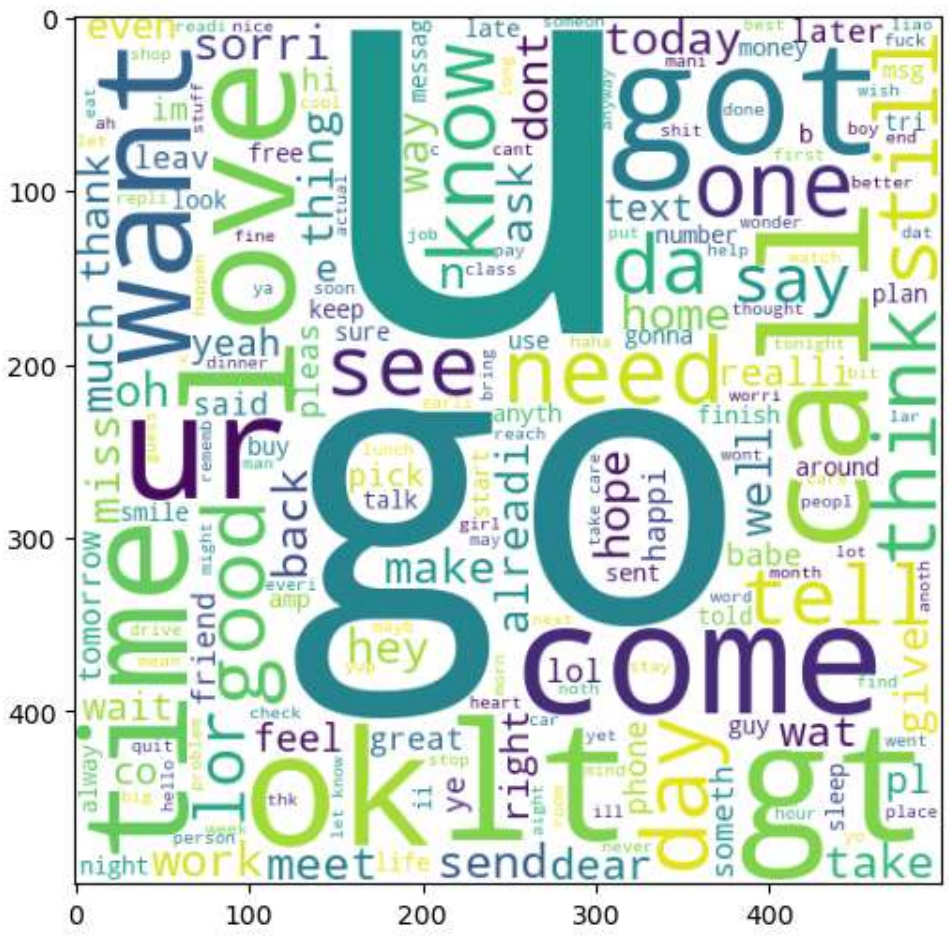
In [37]:
```
plt.imshow(spam_wc)
plt.figure(figsize=(8, 6))
plt.show()
```



<Figure size 800x600 with 0 Axes>

In [38]:
```
ham_wc = wc.generate(data[data['class']==0]['processed_text'].str.cat(sep=" "))
plt.figure(figsize=(8, 6))
plt.imshow(ham_wc)
```

Out[38]: <matplotlib.figure.AxesImage at 0x243233d9010>



In [39]:
```
spam_corpus = []
for x in data[data['class'] == 1]['processed_text'].tolist():
    for y in x.split():
        spam_corpus.append(y)
```

In [40]:
```
len(spam_corpus)
```

Out[40]: 11728

In [41]:
```
'a' in stopwords.words('english')
```

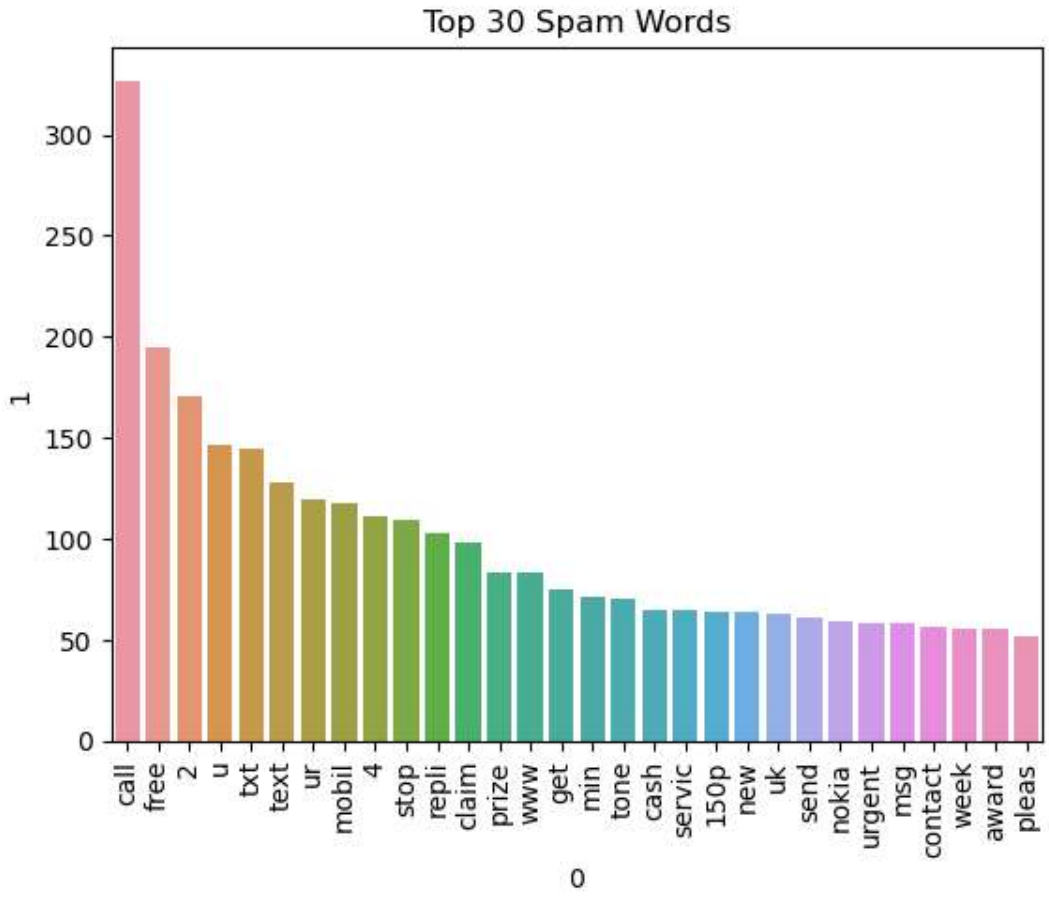
Out[41]: True

In [44]:
```
pd.DataFrame(Counter(spam_corpus).most_common(5))
```

Out[44]:

	0	1
0	call	327
1	free	195
2	2	171
3	u	146
4	txt	145

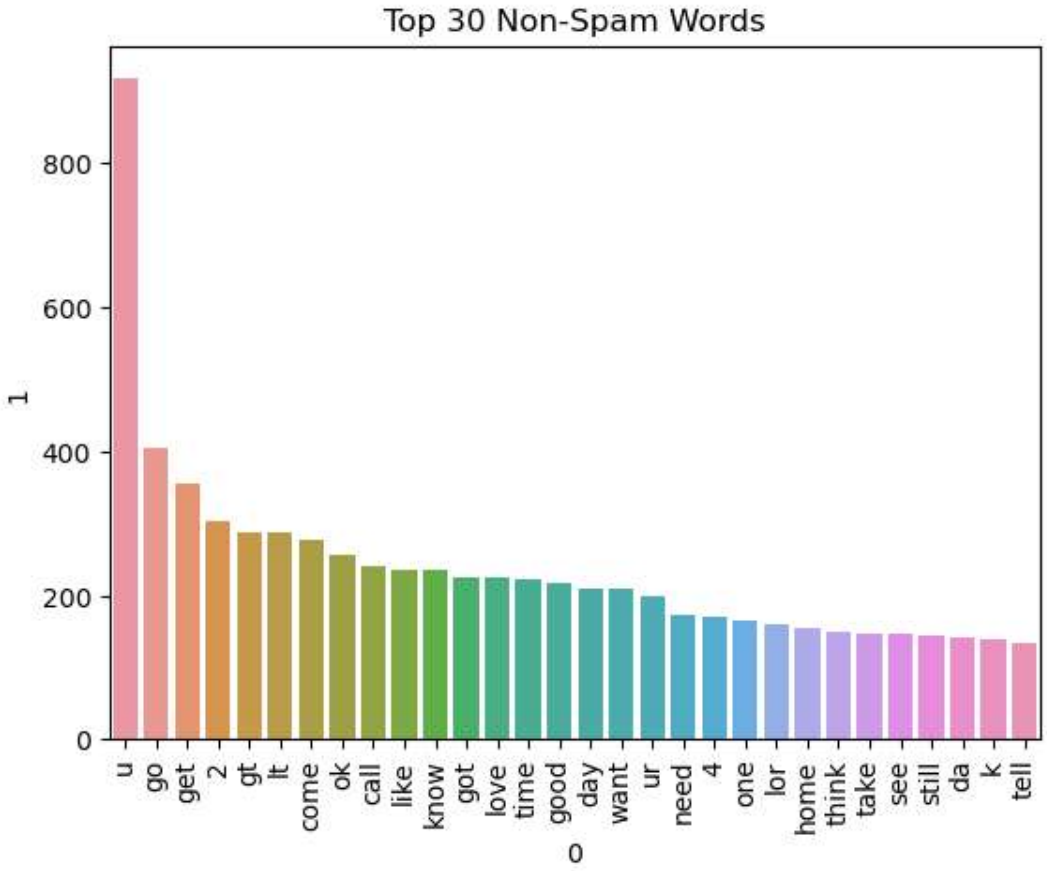
In [43]:
```
from collections import Counter
import seaborn as sns
sns.barplot(x=pd.DataFrame(Counter(spam_corpus).most_common(30))[0], y=pd.DataFrame(Cou
plt.xticks(rotation='vertical')
plt.title("Top 30 Spam Words")
plt.show()
```



In [45]:
```
ham_corpus = []
for x in data[data['class'] == 0]['processed_text'].tolist():
    for y in x.split():
        ham_corpus.append(y)
len(ham_corpus)
```

Out[45]: 36358

In [46]:
```
sns.barplot(x=pd.DataFrame(Counter(ham_corpus).most_common(30))[0], y=pd.DataFrame(Coun
plt.xticks(rotation='vertical')
plt.title("Top 30 Non-Spam Words")
plt.show()
```



Vectorization

In [47]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

In [48]:
```
vec = CountVectorizer()
vector_corpus = vec.fit_transform(data['processed_text'])
vector_corpus.toarray()
```

Out[48]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [49]:
```
col = vec.get_feature_names_out()
col
```

Out[49]:
```
array(['00', '000', '000pe', ..., 'zoom', 'zouk', 'zyada'], dtype=object)
```

In [50]:
```
vector_data = pd.DataFrame(vector_corpus.toarray(), columns = col)
```

In [51]:
```
vector_data.sample(3)
```

Out[51]:

	00	000	000pe	008704050406	0089	0121	01223585236	01223585334	0125698789	02	...	zebra	ze
1245	0	0	0	0	0	0	0	0	0	0	...	0	
2334	0	0	0	0	0	0	0	0	0	0	...	0	
56	0	0	0	0	0	0	0	0	0	0	...	0	

3 rows × 7242 columns

In [52]:
```
vector_data.shape
```

Out[52]:
```
(5169, 7242)
```

Model Training

In [53]:
```
from sklearn.model_selection import train_test_split
X, y = vector_data, data['class']
```

In [54]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, test_size=0.2
```

In [55]:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
```

In [56]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [57]:
```
gnb.fit(X_train, y_train)
gnb_pred = gnb.predict(X_test)
```

In [58]:
```
mnb.fit(X_train, y_train)
mnb_pred = mnb.predict(X_test)
bnb.fit(X_train, y_train)
bnb_pred = bnb.predict(X_test)
```

In [59]:
```
gnb_pred
```

Out[59]:
```
array([1, 0, 0, ..., 0, 0, 1])
```

In [60]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, classifi
```

In [61]:
```
print("GaussianNB :", accuracy_score(gnb_pred, y_test))
print("MultinomialNB :", accuracy_score(mnb_pred, y_test))
print("BernoulliNB :", accuracy_score(bnb_pred, y_test))
```

```
GaussianNB : 0.8713733075435203
MultinomialNB : 0.9816247582205029
BernoulliNB : 0.960348162475822
```

In [62]:
```
print("GaussianNB :", precision_score(gnb_pred, y_test))
print("MultinomialNB :", precision_score(mnb_pred, y_test))
print("BernoulliNB :", precision_score(bnb_pred, y_test))
```

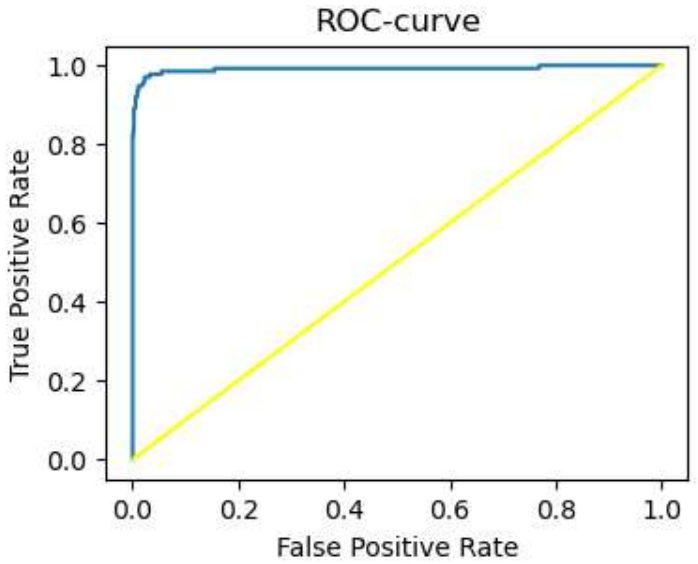
```
GaussianNB : 0.8740740740740741
MultinomialNB : 0.9481481481481482
BernoulliNB : 0.725925925925926
```

AUC-ROC Curve

In [72]:
```
from sklearn.metrics import roc_curve, roc_auc_score
scores = mnb.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, scores)
```

In [95]:
```
plt.figure(figsize=(4, 3))
plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1], color='yellow')
plt.title("ROC-curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

Out[95]:
```
Text(0, 0.5, 'True Positive Rate')
```



Probabilities

In [96]:
```
p = [convert(
    "Hey! We're excited to share a special discount with our loyal customers. Use code S
)]
mnb.predict_proba(vec.transform(p).toarray())[0]
```

Out[96]:
```
array([0.01226813, 0.98773187])
```

Training MultinomialModel on entire Data

In [97]:
```
model = MultinomialNB()
model.fit(X, y)
```

Out[97]:
```
MultinomialNB
MultinomialNB()
```

In [98]:
```
precision_score(model.predict(X), y)
```

Out[98]:
```
0.9693721286370597
```

In [99]:
```
import pickle
```

In [264]:
```
# pickle.dump(model, open('model.pkl', 'wb'))
```

In [268]:
```
# pickle.dump(vec, open('CountVectorizer.pkl', 'wb'))
```

In [269]:
```
# pickle.dump(convert, open('textprocess.pkl', 'wb'))
```

In [100]:
```
m = pickle.load(open('model.pkl', 'rb'))
```

In [102]:
```
v = pickle.load(open('CountVectorizer.pkl', 'rb'))
```

In [103]:
```
tp = pickle.load(open('textprocess.pkl', 'rb'))
```

In [113]:
```
v.transform([tp("the dancing lady your amazing")]).toarray()
```

Out[113]:
```
array([[0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [108]:
```
m.predict_proba(v.transform([tp("You got job offer letter call to this number to confor
```

Out[108]:
```
array([[0.99700304, 0.00299696]])
```

In [2]:

```
import gradio as gr

def greet(name):
    return "Hello " + name + "!"

demo = gr.Interface(fn=greet, inputs="text", outputs="text")

if __name__ == "__main__":
    demo.launch(show_api=False)
```

```
Running on local URL:  http://127.0.0.1:7860 (http://127.0.0.1:7860)

To create a public link, set `share=True` in `launch()`.
```

In [64]:

```
# !pip install gradio
```

In [ ]: