3107 – JAWAHAR ENGINEERING COLLEGE

Subject Title - AI 101- Artificial Intelligence
Project Title – Building a Smarter AI Powered Classifier: Phase-5

Team Member – 310721205001: M. Hariragavan

S.No.	NaanMudhalvan ID	Position	Name	Department
		Faculty Mentor	V. Nivaskumar	
1.	au310721106001	Team Head	Junia Susheela Shalom	ECE
2.	au310721205001	Team member	M. Hariragavan	IT
3.	au310721205002	Team Member	S. Muthunivas Pandi	IT

Project Title-Building a Smarter AI-Powered Spam Classifier

Aim- The objective of this project is to develop a machine learning model that can accurately distinguish between spam and non-spam messages in emails or text-messages based on a set of features such as pattern and probability of different words occurring in spam and ham mail.

Phases of creating an AI Powered Spam Classifier:

1. Data Collection:

> Download a dataset containing labeled examples of spam and non-spam messages from Kaggle.
> Upload the csv file into your Jupyter notebook for further analysis.

2. Data Preprocessing

The text is cleaned and preprocessed. This involves the following:
> Removing special characters.
> Converting text to lowercase.
> Tokenizing the text to individual words.
> Removing stop words and punctuation.
> Lemmatization that involves grouping together different inflected forms of the same word.

3. Feature Extraction

>The tokenized words are converted to numerical features using techniques like TF-IDF (Term Frequency –Inverse Frequency Document Frequency)
> It involves removing specific noisy and less informative terms to enhance the performance of the classifier and decrease feature space dimensionality.

4. Model Selection

> We can experiment with various machine learning algorithms such as Naïve Bayes, Support Vector Machines and more advanced techniques like deep learning using neural networks.
> For this project we implement the Naïve Bayes algorithm.
Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes Theorem and used for solving classification problems.
> It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

5. Evaluation

> The model's performance is measured using metrics like accuracy, precision, recall, and F1-score, Area under Curve, Confusion Matrix and Mean Square Error.
> Model Evaluation is important to assess the efficacy of a model during initial research phases, and it also helps in model monitoring.

6. Iterative Improvement

The model and the experiment are fine-tuned with hyperparameters to improve its accuracy.
The model can be improved by the following:
> Using more training data.
>Reducing or increasing model complexity.
> Applying regularization methods, like Ridge and Lasso regularization.
> In case of Neural networks, adding more dropout layers and early stopping.
> Training the model for more epochs.

STEP 1: Uploading the CSV file into a Jupyter notebook

>First we import the required libraries.
>Secondly we open the csv file using the code: dataset = pd.read_csv('spam.csv')

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
dataset = pd.read_csv('spam.csv')
```

In [3]:
```
dataset.sample(5)
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
1246	ham	I do know what u mean, is the king of not hav...	NaN	NaN	NaN
2339	ham	Cheers for the message Zogtorius. IåÕve been s...	NaN	NaN	NaN
3340	ham	Still i have not checked it da. . .	NaN	NaN	NaN
1351	ham	Yo theres no class tmrw right?	NaN	NaN	NaN
2997	ham	No b4 Thursday	NaN	NaN	NaN

STEP 2: Cleaning the data and preprocessing

>First we converting text to lowercase.
> Secondly, tokenize the text to individual words.
> Then, we remove stop words and punctuation.
>Last but not least, implement Lemmatization (that that involves grouping together different inflected forms of the same word).
> The above processes are performed using the code shown below:

In [54]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('dancing')
```

Out[54]:
```
'danc'
```

In [56]:
```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y=[]
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)
```

In [57]:
```
transform_text('I love the lectures on machine learning')
```

Out[57]:
```
'love lectur machin learn'
```

STEP 3: Feature Extraction

> This step involves converting tokenized words to numerical features.
> Here we use the TF-IDF technique to implement the following lines of code:

In [18]:
```
print(Y.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(5572,)
(4457,)
(1115,)
```

In [21]:
```
feature_extraction = TfidfVectorizer(min_df = 1, stop_words = 'english', lowercase='True')

X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)

Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
```

In [22]:
```
print(X_train)
```

```
3075                  Don know. I did't msg him recently.
1787    Do you know why god created gap between your f...
1614                          Thnx dude. u guys out 2nite?
4304                                       Yup i'm free...
3266    44 7732584351, Do you want a New Nokia 3510i c...
                              ...
789     5 Free Top Polyphonic Tones call 087018728737,...
968     What do u want when i come back?.a beautiful n...
1667    Guess who spent all last night phasing in and ...
3321    Eh sorry leh... I din c ur msg. Not sad alread...
1688    Free Top ringtone -sub to weekly ringtone-get ...
Name: Message, Length: 4457, dtype: object
```

STEP 4: Model Selection

> For this project we implement the Naïve Bayes algorithm.
>Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes Theorem and used for solving classification problems.
> The model can trained as follows:

In [336]:
```
from sklearn.model_selection import train_test_split
```

In [337]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [338]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [339]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [340]:
```
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.874274661508704
[[791 105]
 [ 25 113]]
```

STEP 5: Evaluation and Iterative Improvement of Model

>First, we measure the model's performance using metrics like accuracy, precision, recall, and F1-score, Area under Curve, Confusion Matrix and Mean Square Error.
> Secondly, the model is fine-tuned with hyperparameters to improve its accuracy.
> The above can be implemented by the following lines of code:

In [50]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
print("Precision score: {}". format(precision_score(y_test, prediction)) )
print("Recall score: {}". format(recall_score(y_test, prediction)))
print("F1 score: {}". format(f1_score(y_test, prediction)))
```

```
Accuracy score: 0.97847533632287
Precision score: 0.891156462585034
Recall score: 0.9424460431654677
F1 score: 0.9160839160839161
```

In [54]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
print("Precision score: {}". format(precision_score(y_test, prediction)) )
print("Recall score: {}". format(recall_score(y_test, prediction)))
print("F1 score: {}". format(f1_score(y_test, prediction)))
```

```
Accuracy score: 0.9865470852017937
Precision score: 0.984375
Recall score: 0.9064748201438849
F1 score: 0.9438202247191011
```

Data Cleaning and Preprocessing

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
dataset = pd.read_csv('spam.csv')
```

In [3]:
```
dataset.sample(5)
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
1246	ham	I do know what u mean, is the king of not hav...	NaN	NaN	NaN
2339	ham	Cheers for the message Zogtorius. IåÕve been s...	NaN	NaN	NaN
3340	ham	Still i have not checked it da. . .	NaN	NaN	NaN
1351	ham	Yo theres no class tmrw right?	NaN	NaN	NaN
2997	ham	No b4 Thursday	NaN	NaN	NaN

In [4]:
```
dataset.shape
```

Out[4]:
```
(5572, 5)
```

In [5]:
```
#1.Data Cleaning
#2.EDA
#3.Text Preprocessing
#4.Model Building
#5.Evaluation
#6.Improvement
#7.Website
#8.Deploy
```

1.Data Cleaning

In [6]:
```
dataset.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 [7]:
```
#drop last three columns
dataset.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

In [8]:
```
dataset.sample(5)
```

Out[8]:

	v1	v2

1934	ham	Hope you are having a great day.
196	ham	Did u got that persons story
3346	ham	No problem baby. Is this is a good time to tal...
2049	ham	How much is blackberry bold2 in nigeria.
1220	spam	No. 1 Nokia Tone 4 ur mob every week! Just txt...

In [9]:
```
#rename the columns
dataset.rename(columns={'v1':'type','v2':'text'},inplace=True)
dataset.sample(5)
```

Out[9]:

	type	text
1273	spam	network operator. The service is free. For T &...
3946	ham	Sorry, went to bed early, nightnight
1440	ham	Cool breeze... Bright sun... Fresh flower... T...
3519	ham	No it will reach by 9 only. She telling she wi...
1722	ham	Thought praps you meant another one. Goodo! I'...

In [10]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [11]:
```
encoder.fit_transform(dataset['type'])
```

Out[11]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [12]:
```
dataset['type'] = encoder.fit_transform(dataset['type'])
dataset.head()
```

Out[12]:

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

In [13]:
```
#missing values
dataset.isnull().sum()
```

Out[13]:
```
type    0
text    0
dtype: int64
```

In [14]:
```
#check for duplicate values
dataset.duplicated().sum()
```

Out[14]:
```
403
```

In [15]:
```
#remove duplicates
dataset.drop_duplicates(keep='first')
```

Out[15]:

	type	text
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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5169 rows × 2 columns

In [16]:
```
dataset = dataset.drop_duplicates(keep='first')
```

In [17]:
```
dataset.duplicated().sum()
```

Out[17]: 0

In [18]:
```
dataset.shape
```

Out[18]: (5169, 2)

2.EDA

In [19]:
```
dataset.head()
```

Out[19]:

	type	text
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...

In [20]:
```
dataset.value_counts()
```

Out[20]:
```
type  text
0      &lt;#&gt;  in mca. But not conform.
```

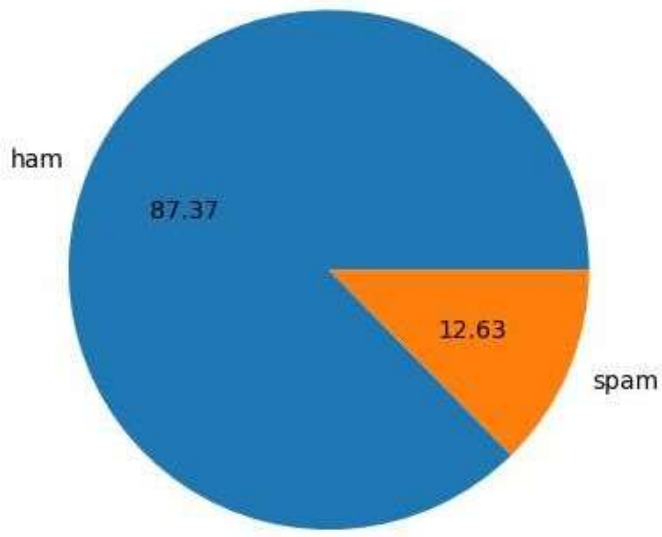
```
1
      Thats cool. i liked your photos. You are very sexy!
1
      That's good, because I need drugs
1
      That's fine, have him give me a call if he knows what he wants or has any questions
1
      That's fine, I'll bitch at you about it later then
1


      I want to send something that can sell fast.  &lt;#&gt; k is not easy money.
1
      I want to see your pretty pussy...
1
      I want to lick your pussy now...
1
      I want to go to perumbavoor
1
1     we tried to contact you re your response to our offer of a new nokia fone and camcorder
hit reply or call 08000930705 for delivery    1
Name: count, Length: 5169, dtype: int64
```

In [21]:
```
dataset['type'].value_counts()
```

Out[21]:
```
type
0    4516
1     653
Name: count, dtype: int64
```

In [22]:
```
import matplotlib.pyplot as plt
plt.pie(dataset['type'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
```



In [23]:
```
#Data is imbalanced
```

In [24]:
```
import nltk
```

In [25]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\shalo\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[25]: True

In [26]:
```
dataset['text'].apply(len)
```

Out[26]:
```
0       111
1        29
2       155
3        49
4        61
       ...
5567    161
5568     37
5569     57
5570    125
5571     26
Name: text, Length: 5169, dtype: int64
```

In [27]:
```
dataset['num_characters'] = dataset['text'].apply(len)
dataset.head()
```

Out[27]:

	type	text	num_characters
0	0	Go until jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives aro...	61

In [28]:
```
# num of words
dataset['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

Out[28]:
```
0       24
1        8
2       37
3       13
4       15
        ..
5567    35
5568     9
5569    15
5570    27
5571     7
Name: text, Length: 5169, dtype: int64
```

In [29]:
```
dataset['num_words']= dataset['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [30]:
```
dataset.head()
```

Out[30]:

	type	text	num_characters	num_words
0	0	Go until jurong point, crazy.. Available only ...	111	24
1	0	Ok lar... Joking wif u oni...	29	8

2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	0	U dun say so early hor... U c already then say...	49	13
4	0	Nah I don't think he goes to usf, he lives aro...	61	15

In [31]:
```
dataset['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

Out[31]:
```
0       2
1       2
2       2
3       1
4       1

5567    4
5568    1
5569    2
5570    1
5571    2
Name: text, Length: 5169, dtype: int64
```

In [32]:
```
dataset['num_sentences'] = dataset['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
dataset.head()
```

Out[32]:

	type	text	num_characters	num words	num_sentences
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

In [33]:
```
dataset[['num_characters','num_words','num_sentences']].describe()
```

Out[33]:

	num_characters	num words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455794	1.965564
std	58.236293	13.324758	1.448541
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

In [34]:
```
#ham
dataset[dataset['type'] ==0][['num_characters','num_words','num_sentences']].describe()
```

Out[34]:

	num_characters	num words	num_sentences
count	4516.000000	4516.000000	4516.000000

mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

In [35]:
```
#spam
dataset[dataset['type'] == 1][['num_characters','num_words','num_sentences']].describe()
```

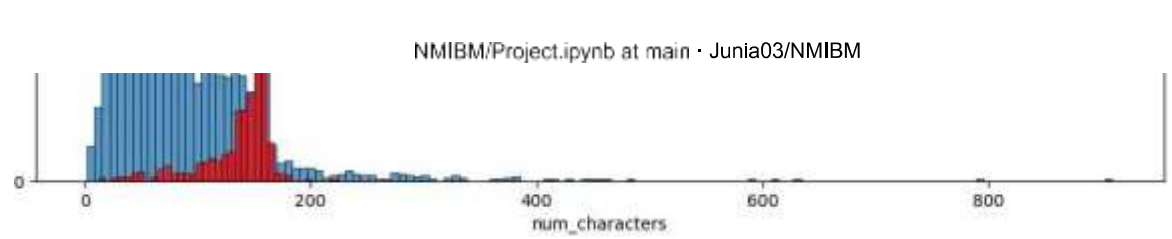
Out[35]:

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
min	13.000000	2.000000	1.000000
25%	132.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	224.000000	46.000000	9.000000

In [36]:
```
import seaborn as sns
```

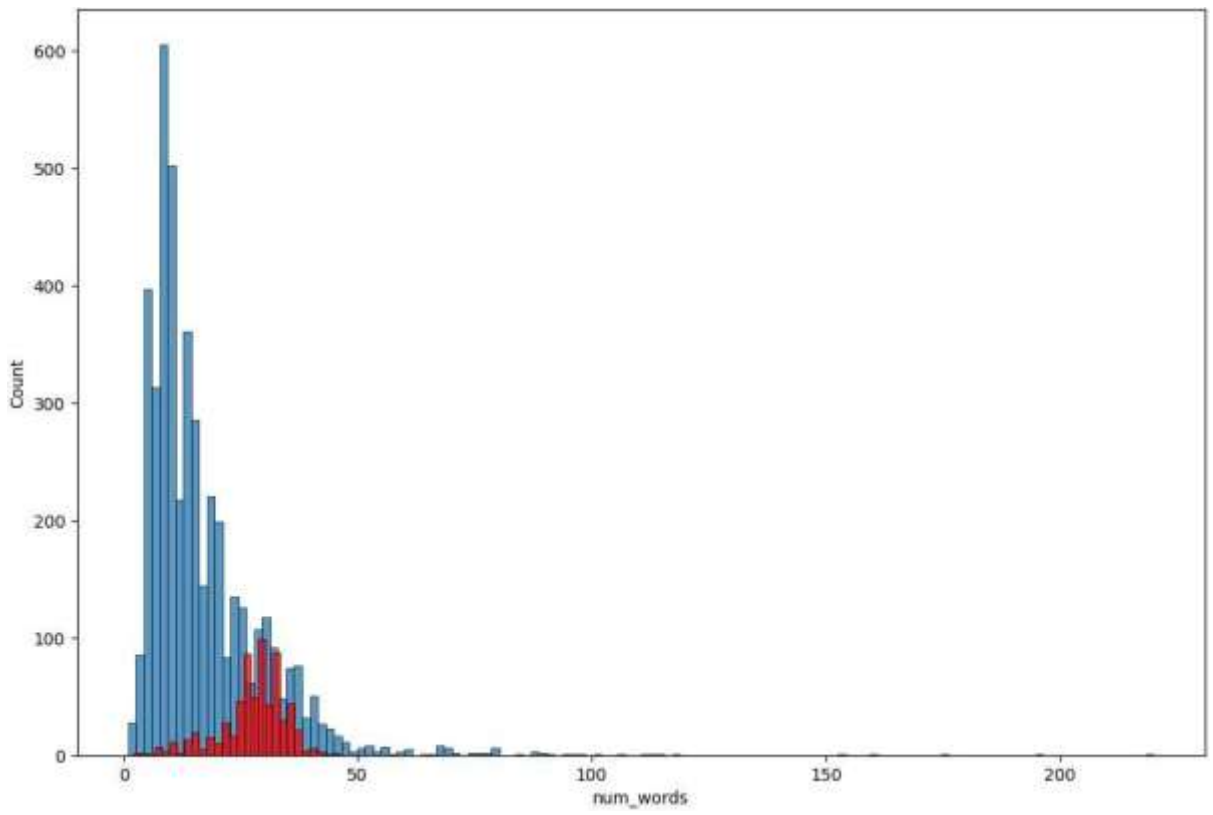
In [37]:
```
plt.figure(figsize = (12,8))
sns.histplot(dataset[dataset['type'] == 0]['num_characters'])
sns.histplot(dataset[dataset['type'] == 1]['num_characters'],color='red')
```

Out[37]:
```
<Axes: xlabel='num_characters', ylabel='Count'>
```



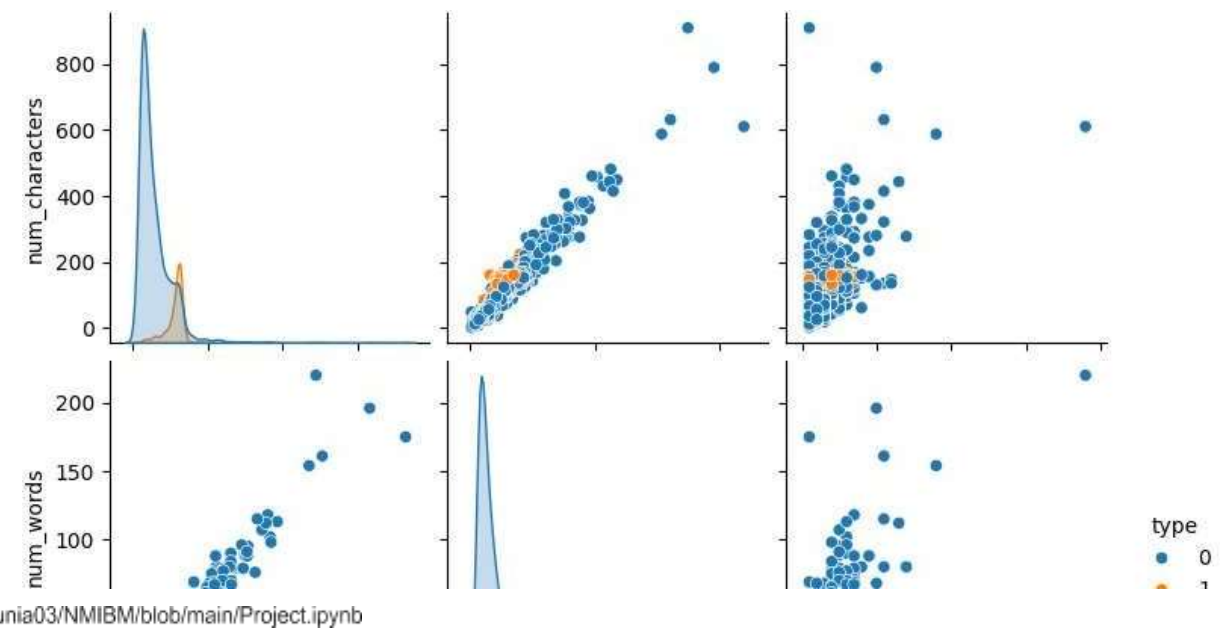
In [38]:
```
plt.figure(figsize = (12,8))
sns.histplot(dataset[dataset['type'] == 0]['num_words'])
sns.histplot(dataset[dataset['type'] == 1]['num_words'],color='red')
```

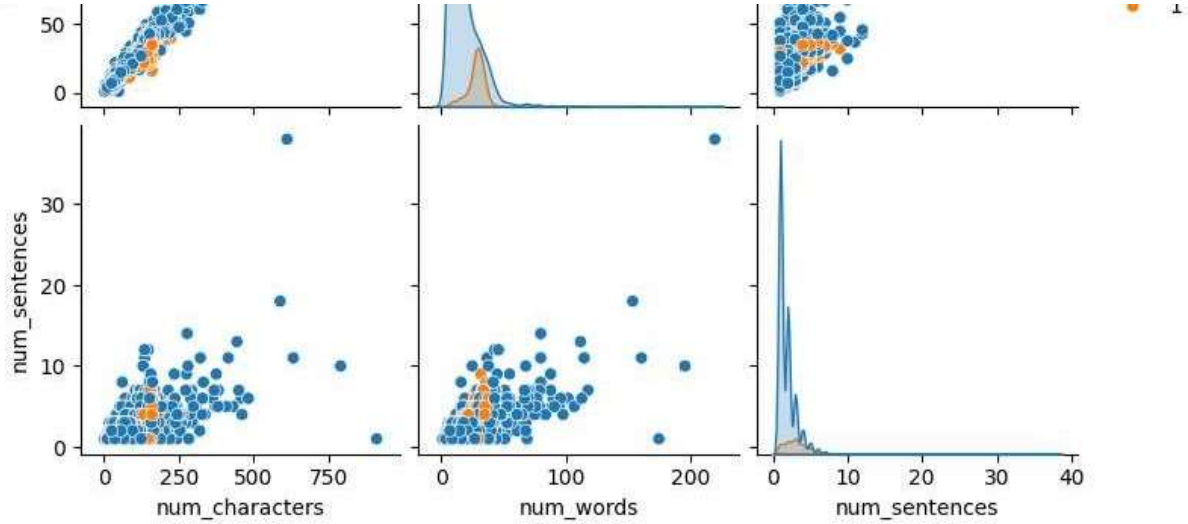
Out[38]: <Axes: xlabel='num_words', ylabel='Count'>



In [39]:
```
sns.pairplot(dataset, hue = 'type')
```

Out[39]: <seaborn.axisgrid.PairGrid at 0x241040d6ed0>





3.Data Preprocessing

In [40]:
```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

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

    return y
```

In [41]:
```
transform_text('Hi How Are You %%')
```

Out[41]:
```
['hi', 'how', 'are', 'you']
```

In [42]:
```
dataset['text'][2000]
```

Out[42]:
```
"But i'll b going 2 sch on mon. My sis need 2 take smth."
```

In [43]:
```
from nltk.corpus import stopwords
stopwords.words('english')
```

Out[43]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
```

```
                'him',
                'his',
                'himself',
                'she',
                "she's",
                'her',
                'hers',
                'herself',
                'it',
                "it's",
                'its',
                'itself',
                'they',
                'them',
                'their',
                'theirs',
                'themselves',
                'what',
                'which',
                'who',
                'whom',
                'this',
                'that',
                "that'll",
                'these',
                'those',
                'am',
                'is',
                'are',
                'was',
                'were',
                'be',
                'been',
                'being',
                'have',
                'has',
                'had',
                'having',
                'do',
                'does',
                'did',
                'doing',
                'a',
                'an',
                'the',
                'and',
                'but',
                'if',
                'or',
                'because',
                'as',
                'until',
                'while',
                'of',
                'at',
                'by',
                'for',
                'with',
                'about',
                'against',
                'between',
                'into',
                'through',
                'during',
                'before',
                'after',
                'above',
                'below',
                'to',
```

```
                'from',
                'up',
                'down',
                'in',
                'out',
                'on',
                'off',
                'over',
                'under',
                'again',
                'further',
                'then',
                'once',
                'here',
                'there',
                'when',
                'where',
                'why',
                'how',
                'all',
                'any',
                'both',
                'each',
                'few',
                'more',
                'most',
                'other',
                'some',
                'such',
                'no',
                'nor',
                'not',
                'only',
                'own',
                'same',
                'so',
                'than',
                'too',
                'very',
                's',
                't',
                'can',
                'will',
                'just',
                'don',
                "don't",
                'should',
                "should've",
                'now',
                'd',
                'll',
                'm',
                'o',
                're',
                've',
                'y',
                'ain',
                'aren',
                "aren't",
                'couldn',
                "couldn't",
                'didn',
                "didn't",
                'doesn',
                "doesn't",
                'hadn',
                "hadn't",
                'hasn',
                "hasn't",
```

```
'haven',
"haven't",
'isn',
"isn't",
'ma',
'mightn',
"mightn't",
'mustn',
"mustn't",
'needn',
"needn't",
'shan',
"shan't",
'shouldn',
"shouldn't",
'wasn',
"wasn't",
'weren',
"weren't",
'won',
"won't",
'wouldn',
"wouldn't"]
```

In [44]:
```
import string
```

In [45]:
```
string.punctuation
```

Out[45]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [54]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('dancing')
```

Out[54]:
```
'danc'
```

In [56]:
```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y=[]
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)
```

In [57]:
```
transform_text('I love the lectures on machine learning')
```

Out[57]:
```
'love lectur machin learn'
```

In [58]:
```
dataset['text'][10]
```

Out[58]:
```
"I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cri
ed enough today."
```

In [59]:
```
dataset['transformed_text'] = dataset['text'].apply(transform_text)
```

In [60]:
```
dataset.head()
```

Out[60]:

	type	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

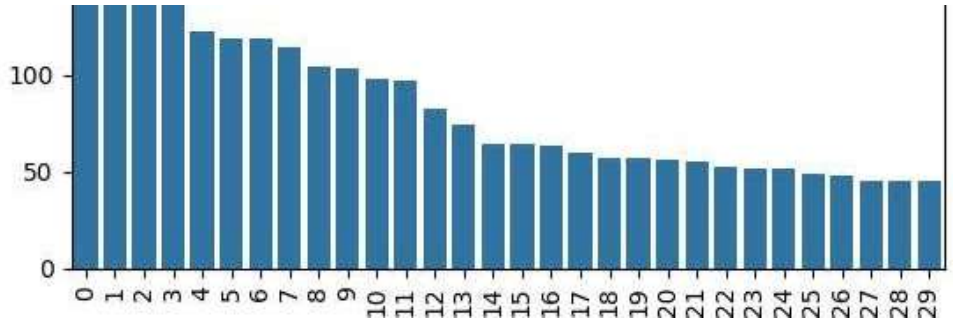
In [61]:
```
spam_corpus = []
for msg in dataset[dataset['type'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

In [64]:
```
len(spam_corpus)
```

Out[64]:
```
9939
```

In [82]:
```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show
```

Out[82]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



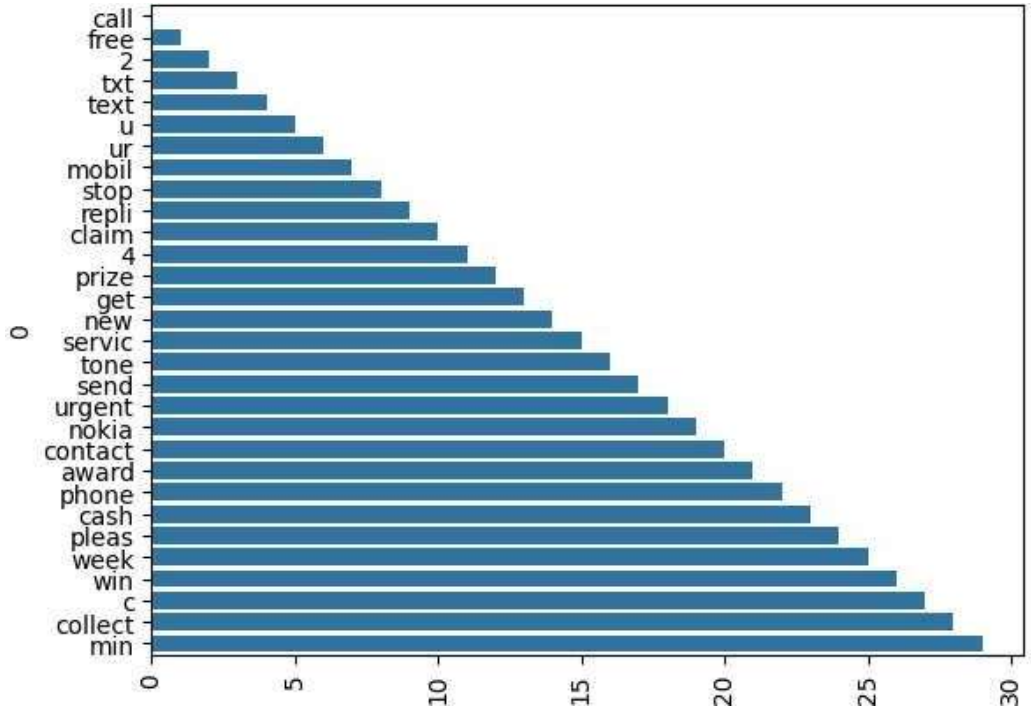
In [84]:
```
ham_corpus = []
for msg in dataset[dataset['type'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [85]:
```
len(ham_corpus)
```

Out[85]: 9939

In [87]:
```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0])
plt.xticks(rotation='vertical')
plt.show
```

Out[87]: <function matplotlib.pyplot.show(close=None, block=None)>



In [ ]:

Model Building

1. Model Building

In [59]:
```
X = tfidf.fit transform(dataset['transformed text']).toarray()
```

In [60]:
```
# appending the num_character column to X
X = np.hstack((X,dataset['num_characters'].values.reshape(-1, 1)))
```

In [61]:
```
X.shape
```

Out[61]: (5169, 3001)

In [62]:
```
y = dataset['type'].values
y
```

Out[62]: array([0, 0, 1, ..., 0, 0, 0])

In [63]:
```
from sklearn.model_selection import train_test_split
```

In [64]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [65]:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score
```

In [66]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [67]:
```
gnb.fit(X_train, y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test, y_pred1))
print(confusion_matrix(y_test, y_pred1))
print(precision_score(y_test, y_pred1))
```

```
0.8907156673114119
[[807  89]
 [ 24 114]]
0.5615763546798029
```

In [68]:
```
mnb.fit(X_train, y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test, y_pred2))
print(confusion_matrix(y_test, y_pred2))
print(precision_score(y_test, y_pred2))
```

```
0.9410058027079303
[[896   0]
 [ 61  77]]
1.0
```

In [69]:
```
bnb.fit(X_train, y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test, y_pred3))
print(confusion_matrix(y_test, y_pred3))
print(precision_score(y_test, y_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

In [70]:
```
## tfidf --> MNB
```

In [71]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from  sklearn.ensemble  import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [72]:
```
svc = SVC(kernel = 'sigmoid', gamma = 1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver = 'liblinear', penalty = 'l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50, random_state=2)
xgb = XGBClassifier(n_estimators=50, random_state=2)
```

In [73]:
```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB' : mnb,
    'DT' : dtc,
    'LR' : lrc,
    'RF' : rfc,
    'Adaboost' : abc,
    'BgC' : bc,
    'ETC' : etc,
    'GBDT' : gbdt,
    'xgb' : xgb
}
```

In [74]:
```
def train_classifier(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    return accuracy, precision
```

In [75]:
```
train_classifier(svc, X_train, y_train, X_test, y_test)
```

```
C:\Users\shalo\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\metrics\_classi
fication.py:1469: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to n
o predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[75]: (0.8665377176015474, 0.0)

In [76]:
```
accuracy_scores = []
precision_scores = []
```

```
for name, clf in clfs.items():
    current_accuracy, current_precision = train_classifier(clf, X_train, y_train, X_test, y_te
    print("For ", name)
    print("Accuracy - ", current_accuracy)
    print("Precision - ", current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
C:\Users\shalo\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\metrics\_classi
fication.py:1469: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to n
o predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
For  SVC
Accuracy -  0.8665377176015474
Precision -  0.0
For  KN
Accuracy -  0.9332688588007737
Precision -  0.822429906542056
For  NB
Accuracy -  0.9410058027079303
Precision -  1.0
For  DT
Accuracy -  0.9439071566731141
Precision  -  0.8773584905660378
For  LR
Accuracy -  0.9613152804642167
Precision  -  0.9622641509433962
For  RF
Accuracy -  0.9690522243713733
Precision  -  0.9818181818181818
For  Adaboost
Accuracy -  0.9642166344294004
Precision  -  0.9316239316239316
For  BgC
Accuracy -  0.9661508704061895
Precision  -  0.8992248062015504
For  ETC
Accuracy -  0.9787234042553191
Precision  -  0.9754098360655737
For  GBDT
Accuracy -  0.9506769825918762
Precision  -  0.9306930693069307
For  xgb
Accuracy -  0.9690522243713733
Precision  -  0.9416666666666667
```

In [77]:
```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores, 'Precision'
```

In [78]:
```
performance_df
```

Out[78]:

	Algorithm	Accuracy	Precision
2	NB	0.941006	1.000000
5	RF	0.969052	0.981818
8	ETC	0.978723	0.975410
4	LR	0.961315	0.962264
10	xgb	0.969052	0.941667
6	Adaboost	0.964217	0.931624
9	GBDT	0.950677	0.930693

7	BgC	0.966151	0.899225
3	DT	0.943907	0.877358
1	KN	0.933269	0.822430
0	SVC	0.866538	0.000000

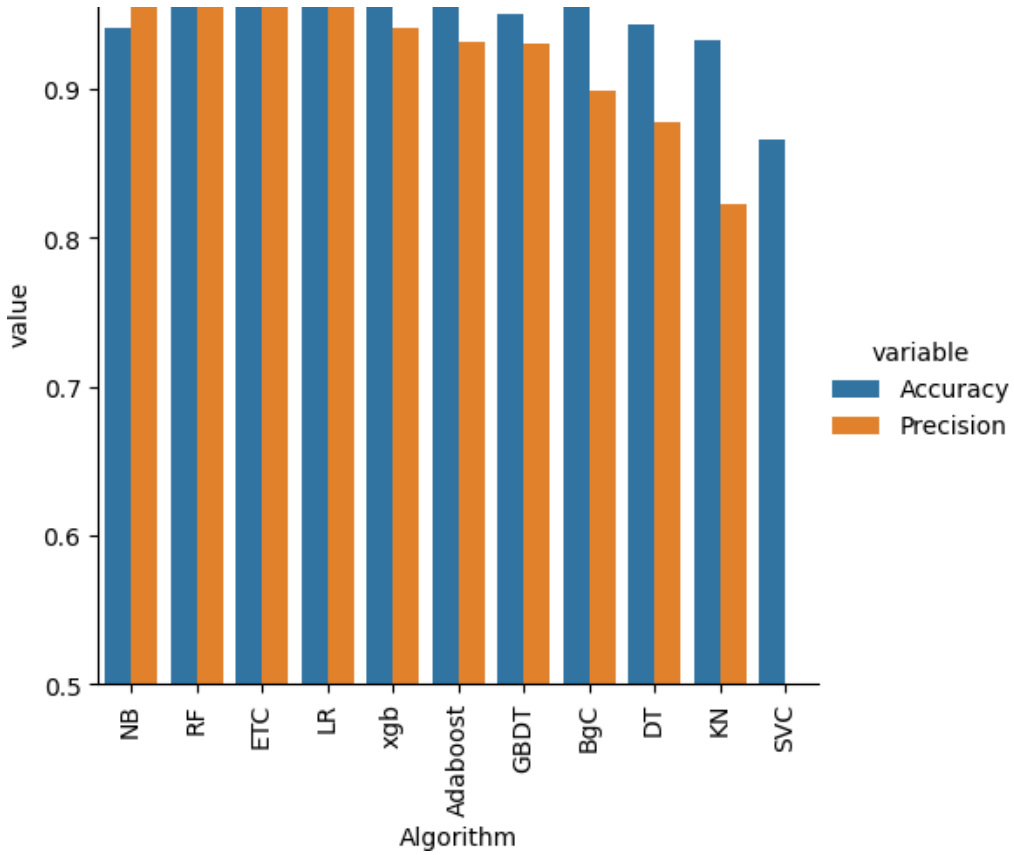
In [79]:
```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
```

In [80]:
```
performance_df1
```

Out[80]:

	Algorithm	variable	value
0	NB	Accuracy	0.941006
1	RF	Accuracy	0.969052
2	ETC	Accuracy	0.978723
3	LR	Accuracy	0.961315
4	xgb	Accuracy	0.969052
5	Adaboost	Accuracy	0.964217
6	GBDT	Accuracy	0.950677
7	BgC	Accuracy	0.966151
8	DT	Accuracy	0.943907
9	KN	Accuracy	0.933269
10	SVC	Accuracy	0.866538
11	NB	Precision	1.000000
12	RF	Precision	0.981818
13	ETC	Precision	0.975410
14	LR	Precision	0.962264
15	xgb	Precision	0.941667
16	Adaboost	Precision	0.931624
17	GBDT	Precision	0.930693
18	BgC	Precision	0.899225
19	DT	Precision	0.877358
20	KN	Precision	0.822430
21	SVC	Precision	0.000000

In [81]:
```
sns.catplot(x = 'Algorithm', y = 'value',
            hue = 'variable', data = performance_df1, kind = 'bar', height = 5)
plt.ylim(0.5, 1.0)
plt.xticks(rotation = 'vertical')
plt.show()
```



In [82]:
```
#model improve
#1. Change the max features parameter of TfIdf
```

In [83]:
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(), 'Accuracy_max_ft_3000': accuracy_scores, 'Pre
```

In [84]:
```
performance_df.merge(temp_df, on='Algorithm')
```

Out[84]:

	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000	Precision_max_ft_3000
0	NB	0.941006	1.000000	0.941006	1.000000
1	RF	0.969052	0.981818	0.969052	0.981818
2	ETC	0.978723	0.975410	0.978723	0.975410
3	LR	0.961315	0.962264	0.961315	0.962264
4	xgb	0.969052	0.941667	0.969052	0.941667
5	Adaboost	0.964217	0.931624	0.964217	0.931624
6	GBDT	0.950677	0.930693	0.950677	0.930693
7	BgC	0.966151	0.899225	0.966151	0.899225
8	DT	0.943907	0.877358	0.943907	0.877358
9	KN	0.933269	0.822430	0.933269	0.822430
10	SVC	0.866538	0.000000	0.866538	0.000000

In [88]:
```
# Voting Classifier
svc = SVC(kernel='sigmoid',gamma = 1.0, probability=True)
mnb = MultinomialNB()
```

```
etc = ExtraTreesClassifier(n_estimators = 50, random_state=2)

from sklearn.ensemble import VotingClassifier
```

In [90]:
```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting = 'soft')
```

In [91]:
```
voting.fit(X_train, y_train)
```

Out[91]:
```
VotingClassifier(estimators=[('svm',
                              SVC(gamma=1.0, kernel='sigmoid',
                                  probability=True)),
                             ('nb', MultinomialNB()),
                             ('et',
                              ExtraTreesClassifier(n_estimators=50,
                                                   random_state=2))],
                 voting='soft')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [92]:
```
y_pred = voting.predict(X_test)
print("Accuracy", accuracy_score(y_test, y_pred))
print("Precision", precision_score(y_test, y_pred))
```

```
Accuracy 0.941972920696325
Precision 1.0
```

In [97]:
```
#Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator = RandomForestClassifier()
```

In [98]:
```
from sklearn.ensemble import StackingClassifier
```

In [99]:
```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
```

In [100…
```
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy", accuracy_score(y_test, y_pred))
print("Precision", precision_score(y_test, y_pred))
```

```
Accuracy 0.9748549323017408
Precision 0.9307692307692308
```

Creating a spam classifier website

Pickle files using the code below:

In [ ]:
```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

Create a python project and import streamlit
Type the code lines given below to create a custom spam classifier website

```
import streamlit as st
import pickle
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()
```

```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

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

    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
```

```
    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

```python
tfidf = pickle.load(open('vectorizer.pkl','rb'))
model = pickle.load(open('model.pkl','rb'))

st.title("Email/SMS Spam CLassifier")

input_sms = st.text_input("Enter the message")

if st.button('Predict'):

    # 1. preprocess
    transformed_sms = transform_text(input_sms)
    # 2. vectorize
    vector_input = tfidf.transform([transformed_sms])
    # 3. predict
    result = model.predict(vector_input)[0]
    # 4. Display
    if result == 1:
        st.header("Spam")
    else:
        st.header("Not Spam")
```

WoooHooo!!!
A AI Powered Spam Classifier has been Created