SpamDetector October,2023

1.1 Objective : Given a labelled dataset containing spam and legitimate messages train a ML model that can identify that a given message is spam or not

#installing required packages #Uncomment following line and run this cell to install required dependencies #pip install wordcloud

In [1]:
```
#importing necessary libraries for reading cleaning and visulaising the data
import pandas as pd
import numpy as np
import nltk
import string
import warnings
warnings.filterwarnings('ignore')
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 Reading data Dataset is in spam.csv file

In [2]:
```
#read the data
df = pd.read_csv('spam.csv',encoding='latin-1')
df
```

Out[2]:

	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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ã¼ 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

5572 rows × 2 columns

1.3 Data Cleaning

In [3]:
```
#renaming columns with meaningful headings
df = df.rename(columns={"Message" : "text", "Category":"label"})
df
```

Out[3]:

	label	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 Ã¼ 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

5572 rows × 2 columns

In [4]:
```
# Encoding text labels with numerical label
# not spam - 0
# spam - 1
df = df.replace(['ham','spam'],[0,1])
```

In [5]:
```
#Uncomment the following line if stopwords is not downloaded
#nltk.download('stopwords')

#Uncomment the following line for toeknizer to work
#nltk.download('punkt')

from nltk.corpus import stopwords

#remove the punctuations
df['text'] = df['text'].str.replace('[^\w\s]','')

#function to remove stopwords
def filter_stopwords(text):
    text = [word for word in text.split() if word.lower() not in stopwords.words('english')]
    return " ".join(text)

df['text'] = df['text'].apply(filter_stopwords) #removing stopwords from each sms

df
```

Out[5]:

	label	text
0	0	Go jurong point crazy Available bugis n great ...
1	0	Ok lar Joking wif u oni
2	1	Free entry 2 wkly comp win FA Cup final tkts 2...
3	0	U dun say early hor U c already say
4	0	Nah dont think goes usf lives around though
...	...	...
5567	1	2nd time tried 2 contact u U Â750 Pound prize ...
5568	0	Ã¼ b going esplanade fr home
5569	0	Pity mood Soany suggestions
5570	0	guy bitching acted like id interested buying s...
5571	0	Rofl true name

5572 rows × 2 columns

1.4 Visualisation

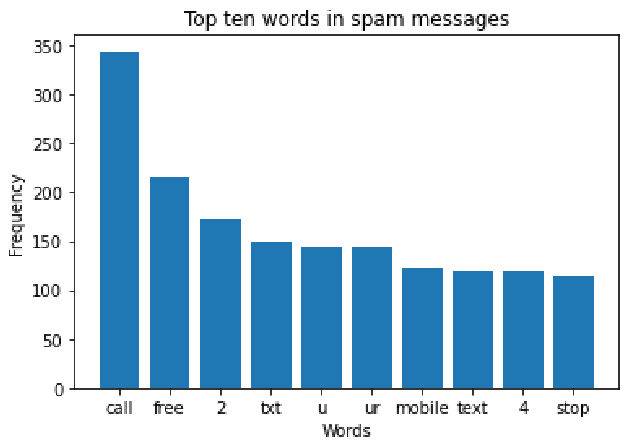
In [6]:
```
# Accumulating all words contained in spam messages
df_spam = df[df['label'] == 1] # all rows with spam
spam_words = ''
for sms in df_spam['text']:
    tokens = nltk.word_tokenize(sms.lower())
    spam_words += ' '.join(tokens) + ' '
print(*spam_words.split()[:300])
```

```
free entry 2 wkly comp win fa cup final tkts 21st may 2005 text fa 87121 receive entry questionstd
txt ratetcs apply 08452810075over18s freemsg hey darling 3 weeks word back id like fun still tb ok
xxx std chgs send â150 rcv winner valued network customer selected receivea â900 prize reward clai
m call 09061701461 claim code kl341 valid 12 hours mobile 11 months u r entitled update latest col
our mobiles camera free call mobile update co free 08002986030 six chances win cash 100 20000 poun
ds txt csh11 send 87575 cost 150pday 6days 16 tsandcs apply reply hl 4 info urgent 1 week free mem
bership â100000 prize jackpot txt word claim 81010 tc wwwdbuknet lccltd pobox 4403ldnw1a7rw18 xxxm
obilemovieclub use credit click wap link next txt message click httpwap xxxmobilemovieclubcomnqjkg
ighjjgcbl england v macedonia dont miss goalsteam news txt ur national team 87077 eg england 87077
trywales scotland 4txtâº120 poboxox36504w45wq 16 thanks subscription ringtone uk mobile charged â5
month please confirm replying yes reply charged 07732584351 rodger burns msg tried call reply sms
free nokia mobile free camcorder please call 08000930705 delivery tomorrow sms ac sptv new jersey
devils detroit red wings play ice hockey correct incorrect end reply end sptv congrats 1 year spec
ial cinema pass 2 call 09061209465 c suprman v matrix3 starwars3 etc 4 free bx420ip45we 150pm dont
miss valued customer pleased advise following recent review mob awarded â1500 bonus prize call 090
66364589 urgent ur awarded complimentary trip eurodisinc trav acoentry41 â1000 claim txt dis 87121
186â150morefrmmob shracomorsglsuplt10 ls1 3aj hear new divorce barbie comes kens stuff please call
customer service representative 0800 169 6031 10am9pm guaranteed â1000 cash â5000 prize free ringt
one waiting collected simply text password mix 85069 verify get usher britney fml po box 5249 mk17
92h 450ppw 16 gent trying contact last weekends draw shows â1000 prize
```

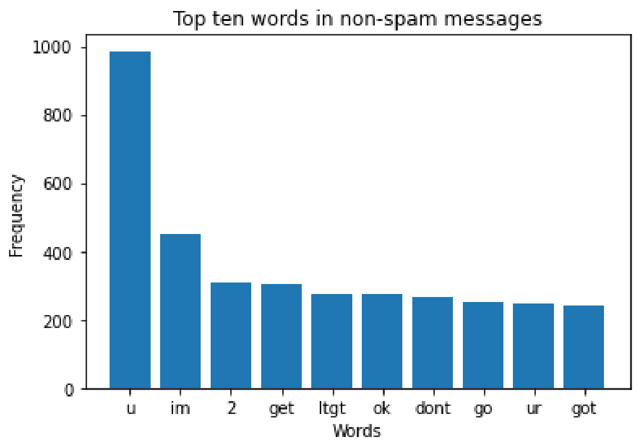
In [13]:
```
# Accumulating all words contained in legitimate messages
df_legit = df[df['label'] == 0] # all rows with no spam
legit_words = ''
for sms in df_legit['text']:
    tokens = nltk.word_tokenize(sms.lower())
    legit_words += ' '.join(tokens) + ' '
print(*legit_words.split()[:300])
```

```
go jurong point crazy available bugis n great world la e buffet cine got amore wat ok lar joking w
if u oni u dun say early hor u c already say nah dont think goes usf lives around though even brot
her like speak treat like aids patent per request melle melle oru minnaminunginte nurungu vettam s
et callertune callers press 9 copy friends callertune im gon na home soon dont want talk stuff any
more tonight k ive cried enough today ive searching right words thank breather promise wont take h
elp granted fulfil promise wonderful blessing times date sunday oh kim watching eh u remember 2 sp
ell name yes v naughty make v wet fine thatâs way u feel thatâs way gota b seriously spell name iâ
m going try 2 months ha ha joking ã¼ pay first lar da stock comin aft finish lunch go str lor ard
3 smth lor u finish ur lunch already ffffffffff alright way meet sooner forced eat slice im really
hungry tho sucks mark getting worried knows im sick turn pizza lol lol always convincing catch bus
frying egg make tea eating moms left dinner feel love im back amp packing car ill let know theres
room ahhh work vaguely remember feel like lol wait thats still clear sure sarcastic thats x doesnt
want live us yeah got 2 v apologetic n fallen actin like spoilt child got caught till 2 wont go ba
dly cheers k tell anything fear fainting housework quick cuppa yup ok go home look timings msg ã¼
xuhui going learn 2nd may lesson 8am oops ill let know roommates done see letter b car anything lo
r u decide hello hows saturday go texting see youd decided anything tomo im trying invite anything
pls go ahead watts wanted sure great weekend abiola forget
```

In [8]:
```
frq_dist = nltk.FreqDist(nltk.tokenize.word_tokenize(spam_words))
top_ten = frq_dist.most_common(10)
x = [wf_pair[0] for wf_pair in top_ten]
y = [wf_pair[1] for wf_pair in top_ten]
plt.bar(x,y)
plt.title('Top ten words in spam messages')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.show()
```



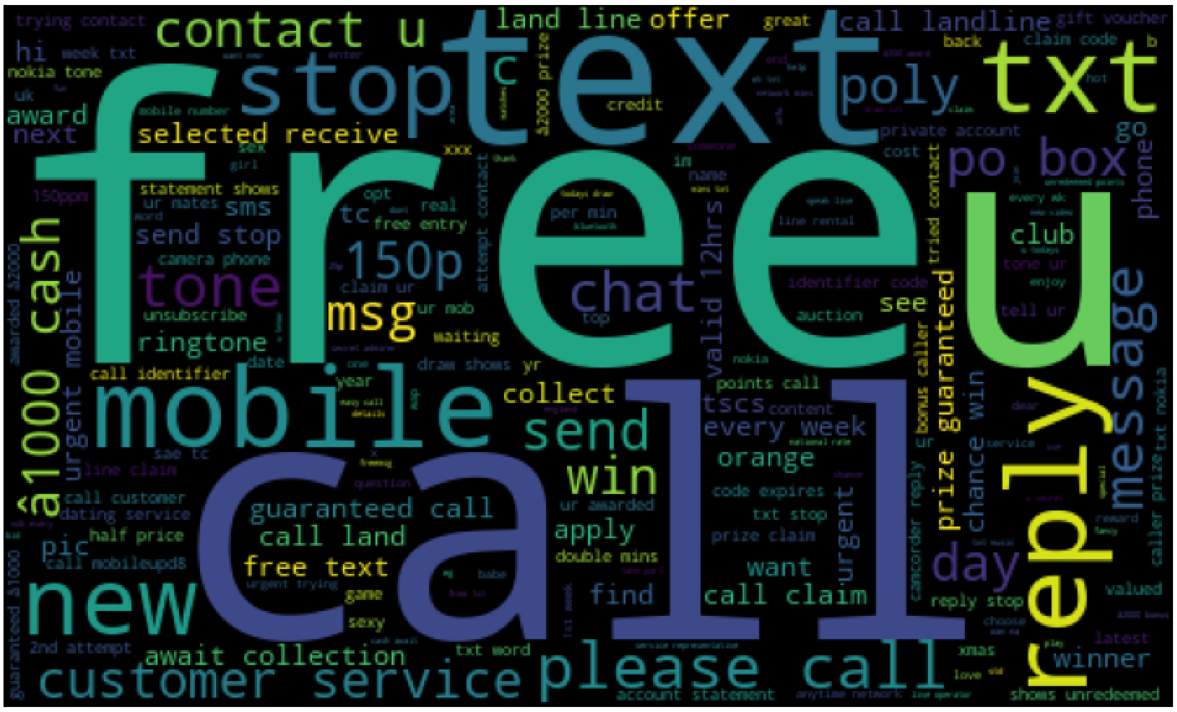
In [14]:
```
frq_dist = nltk.FreqDist(nltk.tokenize.word_tokenize(legit_words))
top_ten = frq_dist.most_common(10)
x = [wf_pair[0] for wf_pair in top_ten]
y = [wf_pair[1] for wf_pair in top_ten]
plt.title('Top ten words in non-spam messages')
plt.bar(x,y)
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.show()
```



In [15]:
```
#creating word clouds for visualisation
spam_wordcloud = WordCloud(width=500, height=300).generate(spam_words)
legit_wordcloud = WordCloud(width=500, height=300).generate(legit_words)
```

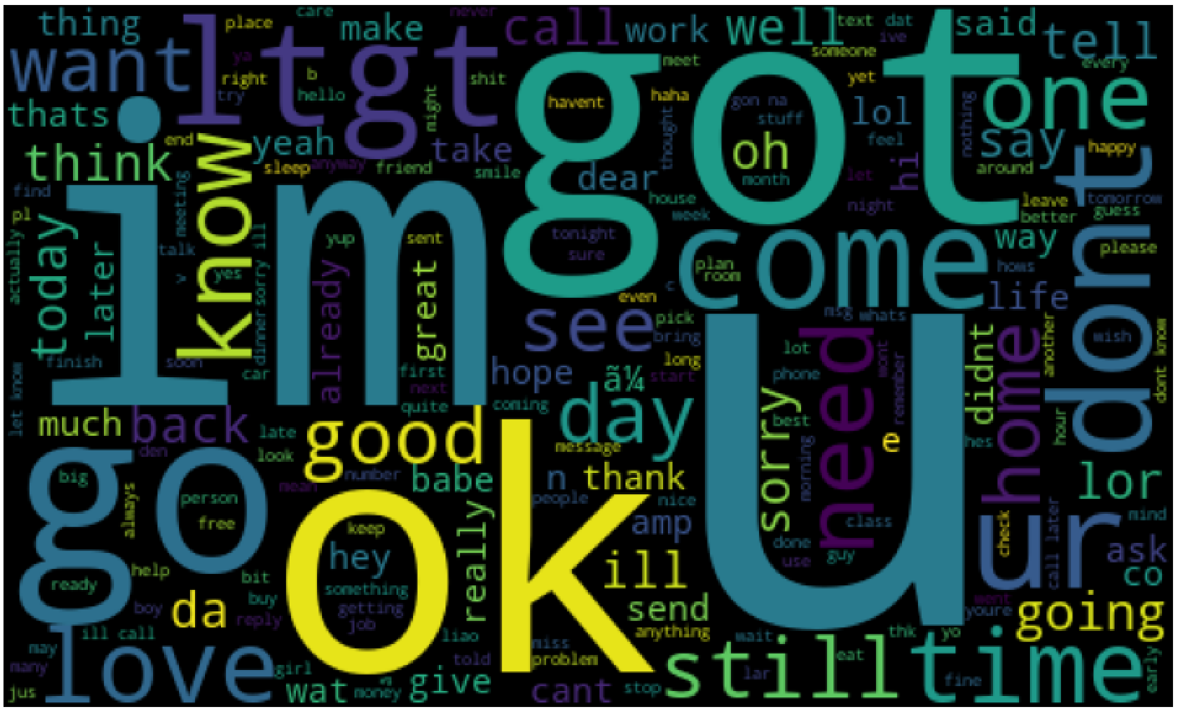
In [16]:
```python
#Displaying word cloud for spam words
plt.figure( figsize=(10,8), facecolor='w')
plt.title('Word cloud for spam-words')
plt.imshow(spam_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

Word cloud for spam-words



In [17]:
```python
#Displaying word cloud for non-spam words
plt.figure( figsize=(10,8), facecolor='w')
plt.title('Word cloud for non-spam words')
plt.imshow(legit_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

Word cloud for non-spam words



1.5 Vectorizing

Converting words to numerical data Count vectorization is basically a way to turn words in documents into numbers, making it easier for computers to understand and analyze the text.

Steps to create count vector

- Vocabulary Creation: Make a list of unique words from the documents.
- Word Counting: Count how many times each word appears in each document.
- Vector Creation: Turn word counts into numbers, creating a table (matrix).

In [18]:
```
#converting text to count vector
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
vectorizer = CountVectorizer()
x = vectorizer.fit_transform(df['text'])
#splitting data into train and test set
X_train, X_test, y_train, y_test = train_test_split(x, df['label'], test_size=0.25, random_state=42
```

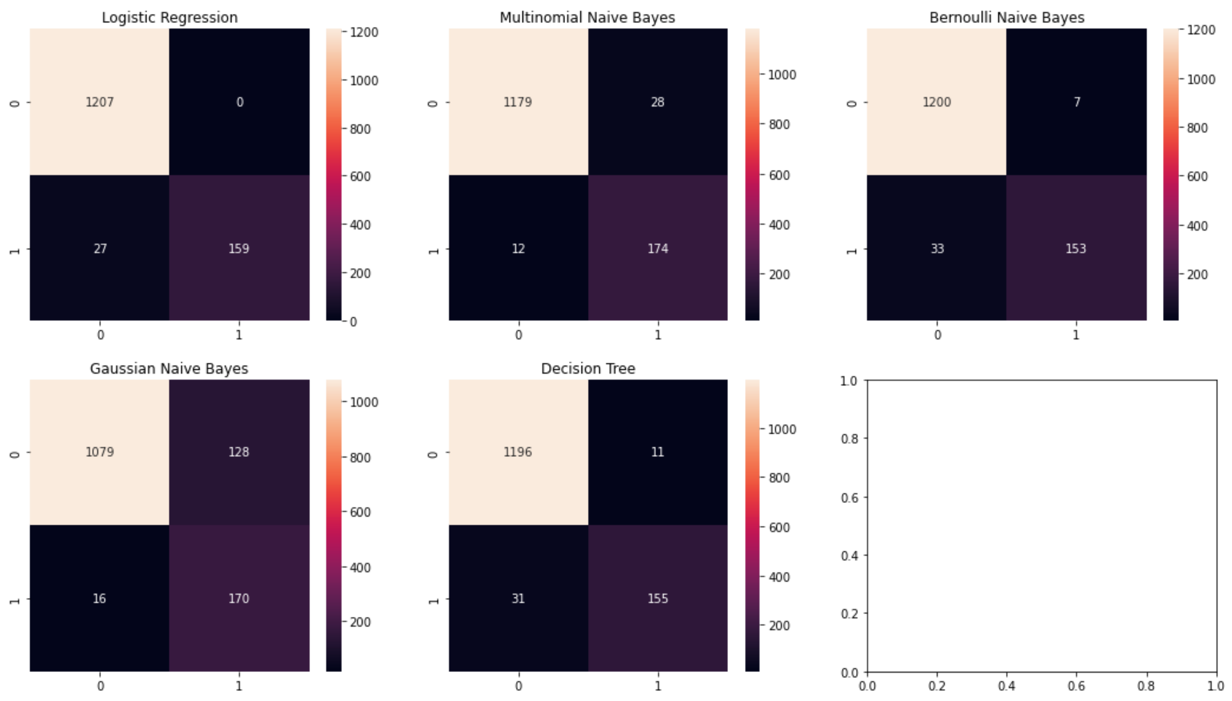
In [19]:
```
#import classification models
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB, BernoulliNB, GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [20]:
```
#initialize multiple classification models
lr = LogisticRegression()
mnb = MultinomialNB()
bnb = BernoulliNB()
gnb = GaussianNB()
dtc = DecisionTreeClassifier(min_samples_split=7, random_state=111)

models = [lr, mnb, bnb, gnb, dtc]

#trains given model on training data and prints accuracy score and confusion matrix

def use_model(model,X_train,X_test,y_train,y_test):
    model.fit(X_train.toarray(), y_train)
    y_pred = model.predict(X_test.toarray())
    acc = accuracy_score(y_test,y_pred)
    print(f'Accuracy of model {model} is {acc*100:.2f}%')
    cm = confusion_matrix(y_test,y_pred)
    print('Confusion matrix: ')
    print(cm)
    return cm
```

In [21]:
```
#train each model on training data and draw confusion matrix
cms=[]
fig, axes = plt.subplots(2,3, figsize=(18, 10))
for model in models:
    cm = use_model(model,X_train,X_test,y_train,y_test)
    cms.append(cm)
sns.heatmap(cms[0], annot = True,fmt = ".0f", ax=axes[0, 0]).set_title('Logistic Regression')

sns.heatmap(cms[1], annot = True,fmt = ".0f", ax=axes[0, 1]).set_title('Multinomial Naive Bayes')

sns.heatmap(cms[2], annot = True,fmt = ".0f", ax=axes[0, 2]).set_title('Bernoulli Naive Bayes')

sns.heatmap(cms[3], annot = True,fmt = ".0f", ax=axes[1, 0]).set_title('Gaussian Naive Bayes')

sns.heatmap(cms[4], annot = True,fmt = ".0f", ax=axes[1, 1]).set_title('Decision Tree')
```

```
Accuracy of model LogisticRegression() is 98.06%
Confusion matrix:
[[1207    0]
 [  27  159]]
Accuracy of model MultinomialNB() is 97.13%
Confusion matrix:
[[1179   28]
 [  12  174]]
Accuracy of model BernoulliNB() is 97.13%
Confusion matrix:
[[1200    7]
 [  33  153]]
Accuracy of model GaussianNB() is 89.66%
Confusion matrix:
[[1079  128]
 [  16  170]]
Accuracy of model DecisionTreeClassifier(min_samples_split=7, random_state=111) is 96.98%
Confusion matrix:
[[1196   11]
 [  31  155]]
```

Out[21]:
```
Text(0.5, 1.0, 'Decision Tree')
```



In [22]:
```
#checks if message is spam or not using given model and using naive bayes as default model

def is_spam(text,model_name='naive_bayes'):
    model = None
    if model_name == 'logistic_regression':
        model = models[0]
    elif model_name == 'naive_bayes':
        model = models[1]
    elif model_name == 'bernoulli_naive_bayes':
        model = models[2]
    elif model_name == 'gaussian_naive_bayes':
        model = models[3]
    elif model_name == 'decision_tree':
        model = models[4]
    if model == None:
        print('Invalid model name')
        return
    is_spam = model.predict(vectorizer.transform([text.lower()]).toarray())[0]== 1
    return is_spam
```

1.6 Examples

In [23]:
```
is_spam('Congratulations, You have won in lucky draw!')
```

Out[23]:
```
True
```

In [24]:
```
is_spam('hello there')
```

Out[24]:
```
False
```

In [ ]:
```

```