In [86]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
%matplotlib inline
```

In [49]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

In [50]:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, precision_score, confusion_matrix, recal
```

In [51]:
```
import nltk
from nltk import word_tokenize
import string, re
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')
from nltk.stem import LancasterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
nltk.download('wordnet')
```

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

Out[51]: True

In [52]:
```
df = pd.read_csv('spam.csv')
```

In [53]:
```
df.head()
```

Out[53]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [54]:
```
df.columns
```

Out[54]:
```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

In [55]:
```
df.sample(5)
```

Out[55]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
4862	spam	Bored housewives! Chat n date now! 0871750.77....	NaN	NaN	NaN
4317	ham	S...i will take mokka players only:)	NaN	NaN	NaN
1795	ham	I hope your alright babe? I worry that you mig...	NaN	NaN	NaN
4684	ham	Alright we'll bring it to you, see you in like...	NaN	NaN	NaN
5278	spam	URGENT! Your Mobile number has been awarded wi...	NaN	NaN	NaN

In [56]:
```
df.shape
```

Out[56]:
```
(5572, 5)
```

Data Cleaning

In [57]:
```
df.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 [58]:
```
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

In [59]:
```
df.sample(5)
```

Out[59]:

	v1	v2
3924	ham	As if i wasn't having enough trouble sleeping.
443	ham	Sorry, I'll call later
879	spam	U have a Secret Admirer who is looking 2 make ...
1978	ham	No I'm in the same boat. Still here at my moms...
3162	spam	This is the 2nd time we have tried to contact ...

In [60]:

```
df.rename(columns={'v1': 'target', 'v2': 'text'}, inplace=True)
df.sample(5)
```

Out[60]:

	target	text
4750	spam	Your weekly Cool-Mob tones are ready to downlo...
2368	ham	If you r @ home then come down within 5 min
5485	ham	Also fuck you and your family for going to rho...
32	ham	K tell me anything about you.
3991	ham	Dizzamn, aight I'll ask my suitemates when I g...

In [61]:

```
encoder = LabelEncoder()
```

In [62]:

```
df['target'] = encoder.fit_transform(df['target'])
```

In [63]:

```
df.head()
```

Out[63]:

	target	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 [64]:

```
df.isnull().sum()
```

Out[64]:

```
target    0
text      0
dtype: int64
```

In [65]:

```
df.duplicated().sum()
```

Out[65]:

```
403
```

In [66]:
```
df = df.drop_duplicates(keep='first')
```

In [67]:
```
df.duplicated().sum()
```

Out[67]:
```
0
```

In [68]:
```
df.shape
```

Out[68]:
```
(5169, 2)
```

Exploratory Data Analysis (EDA)

In [69]:
```
df.head()
```

Out[69]:

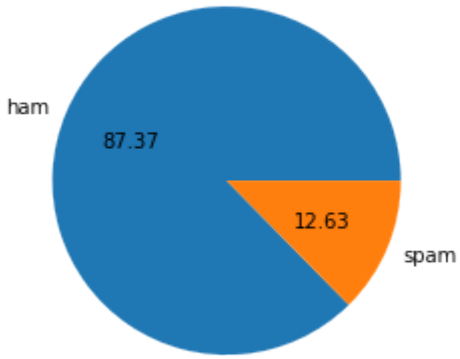
	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

In [70]:
```
df['target'].value_counts()
```

Out[70]:
```
0    4516
1     653
Name: target, dtype: int64
```

In [71]:
```
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
```

Out[71]:
```
([<matplotlib.patches.Wedge at 0x1b8616a0d30>,
  <matplotlib.patches.Wedge at 0x1b8616af460>],
 [Text(-1.0144997251399075, 0.4251944351600247, 'ham'),
  Text(1.014499764949479, -0.4251943401757036, 'spam')],
 [Text(-0.5533634864399495, 0.23192423736001344, '87.37'),
  Text(0.5533635081542612, -0.23192418555038377, '12.63')])
```



Observe that the proportion between ham and spam drastically increases with inclusion of numbers in the text. Therefore, Data is imbalanced.

In [72]:
```
nltk.download('punkt')
```

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

Out[72]: True

In [73]:
```
df['num_characters'] = df['text'].apply(len)
```

In [74]:
```
df.head()
```

Out[74]:

	target	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 [75]:
```
#number of words
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [76]:
```
df.head()
```

Out[76]:

	target	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 [77]:
```
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

In [78]:
```
df.head()
```

Out[78]:

	target	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

	target	text	num_characters	num_words	num_sentences
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 [79]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[79]:

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.923776	18.456375	1.962275
std	58.174846	13.323322	1.433892
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 [80]:
```
#ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

Out[80]:

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.456820	17.123339	1.815545
std	56.356802	13.491315	1.364098
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 [81]:
```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

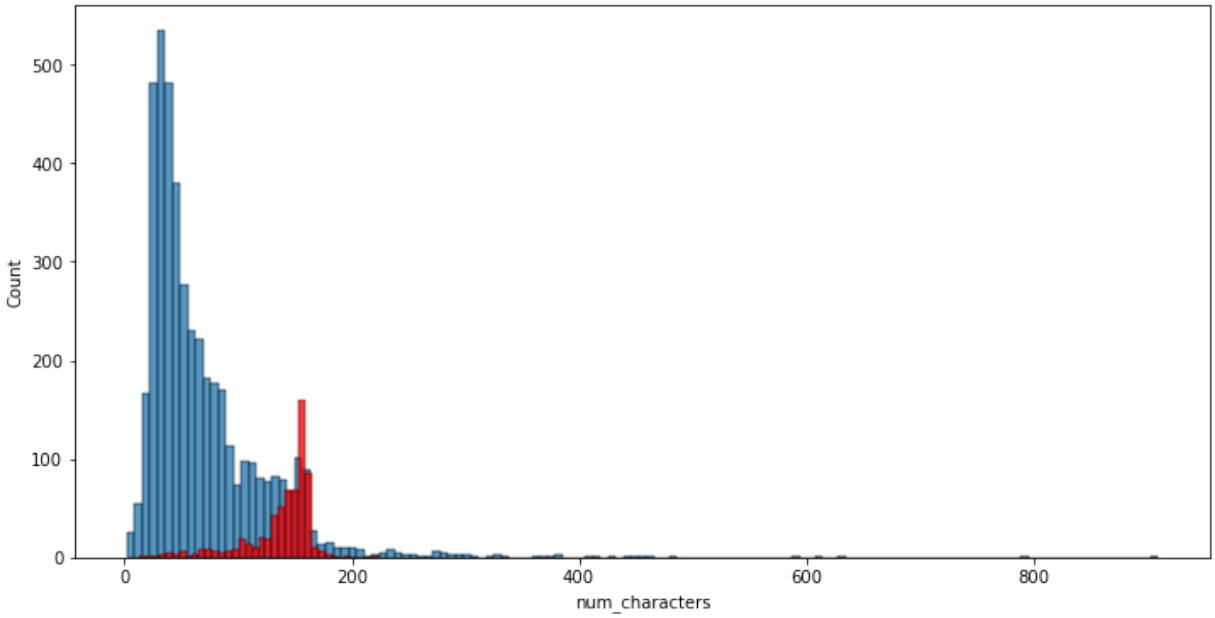
Out[81]:

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.479326	27.675345	2.977029
std	30.014336	7.011513	1.493676
min	13.000000	2.000000	1.000000
25%	131.000000	25.000000	2.000000
50%	148.000000	29.000000	3.000000

	num_characters	num_words	num_sentences
75%	157.000000	32.000000	4.000000
max	223.000000	46.000000	9.000000

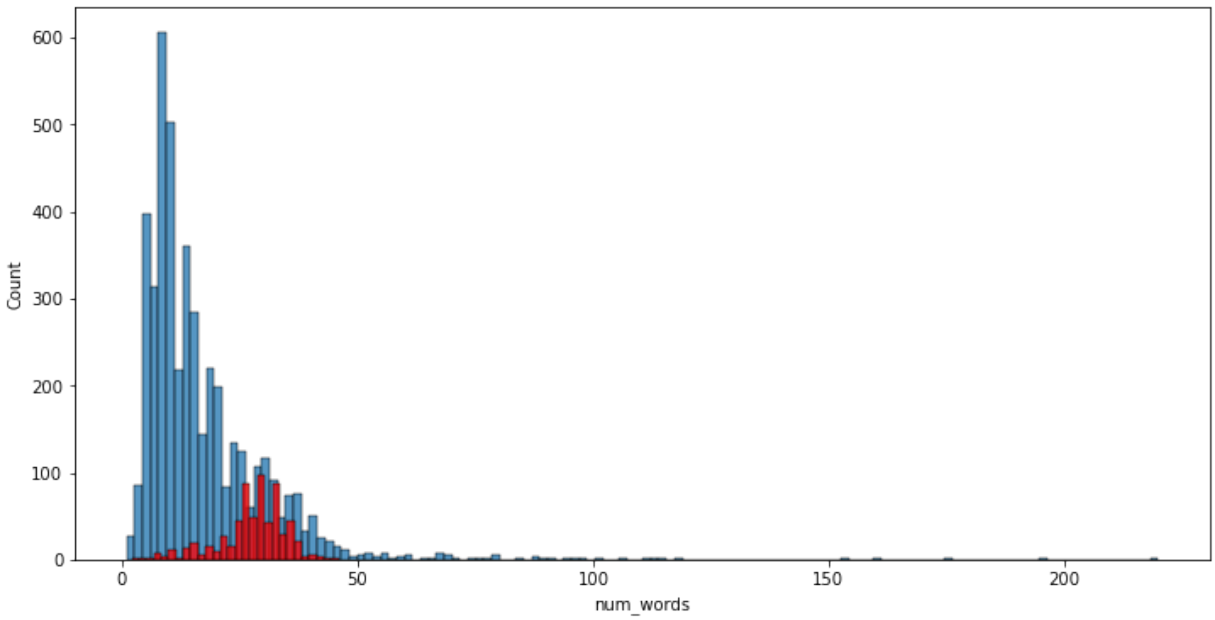
In [82]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

Out[82]: <AxesSubplot:xlabel='num_characters', ylabel='Count'>



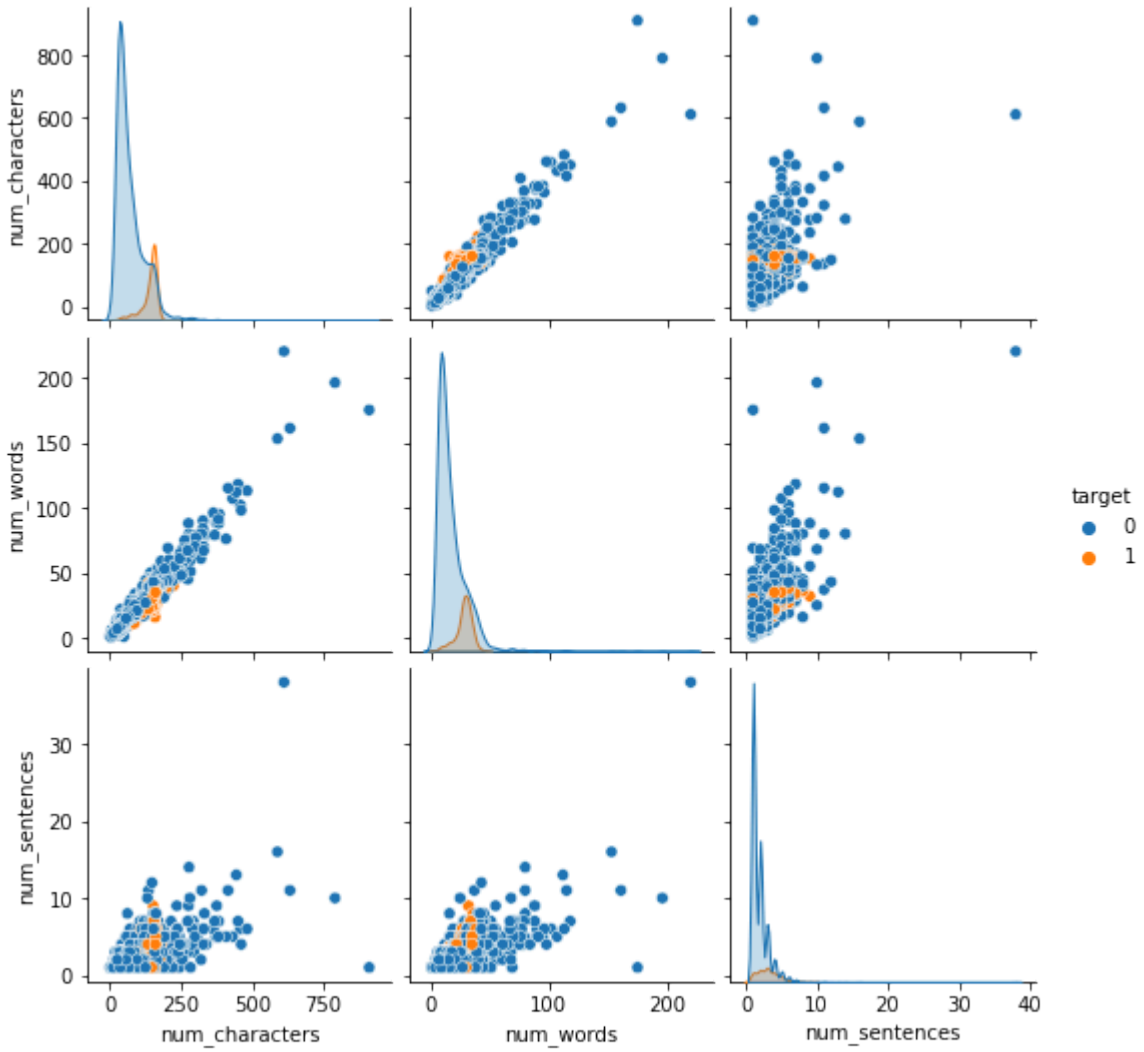
In [83]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

Out[83]: <AxesSubplot:xlabel='num_words', ylabel='Count'>



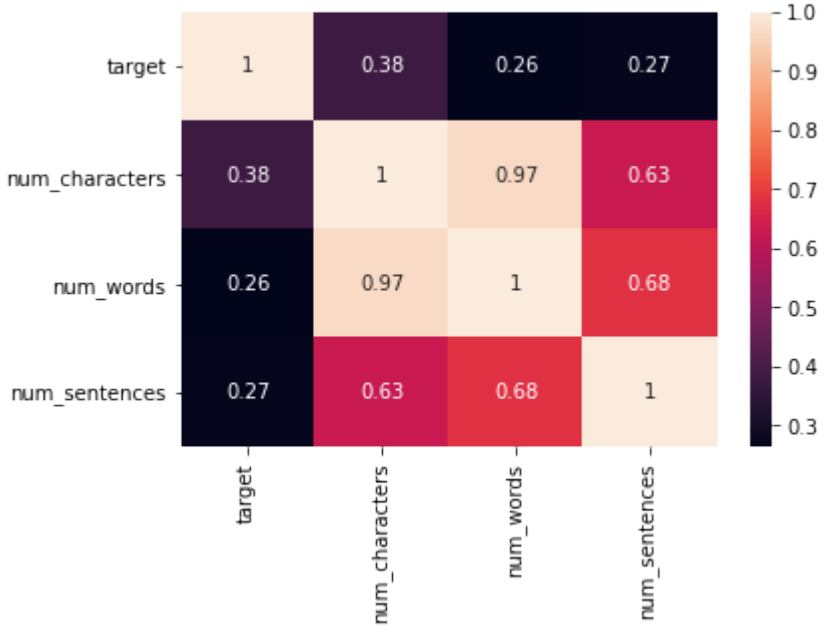
In [84]:
```
sns.pairplot(df,hue='target')
```

Out[84]: <seaborn.axisgrid.PairGrid at 0x1b8602fbee0>



In [85]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[85]: <AxesSubplot:>



Data Preprocessing

- Lower case
- Tokenization
- Removing special characters
- Removing stop words and punctuation
- Stemming

In [86]:
```
from nltk.corpus import stopwords
stopwords.words('english')
```

Out[86]:
```
['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 [87]:

```
import string
string.punctuation
```

Out[87]:

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

In [88]:

```
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 [95]:
```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff any
```

Out[95]:
```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

In [92]:
```
df['text'][10]
```

Out[92]:
```
"I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k?
I've cried enough today."
```

In [94]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

Out[94]:
```
'love'
```

In [96]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

In [97]:
```
df.head()
```

Out[97]:

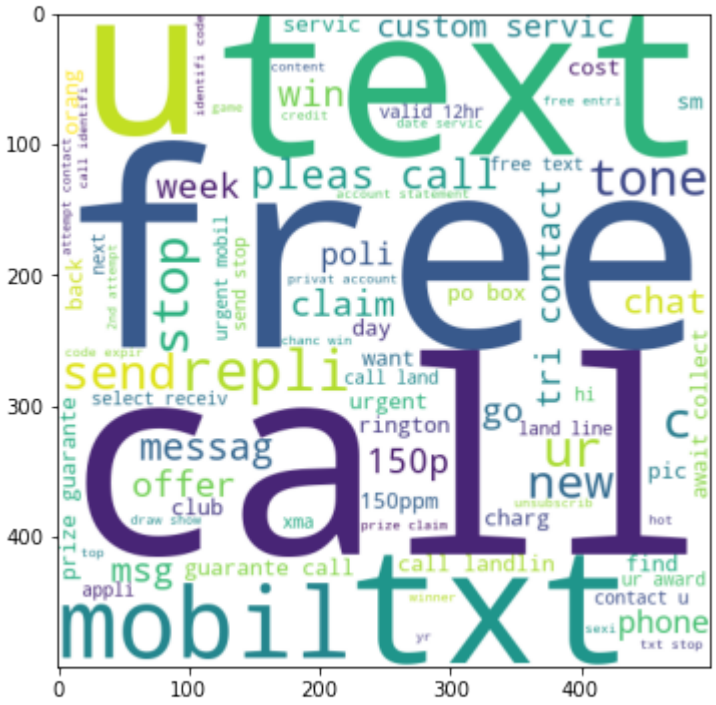
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	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 [98]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

In [99]:
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

In [100…
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

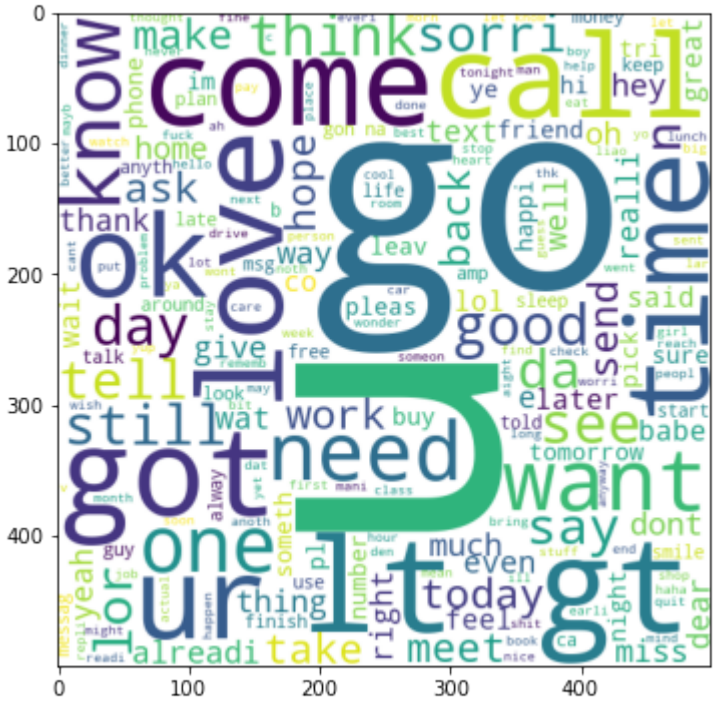
Out[100…
```
<matplotlib.image.AxesImage at 0x1b8626514c0>
```



In [101…
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

In [102…
```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

Out[102…
```
<matplotlib.image.AxesImage at 0x1b863bcf7c0>
```



In [103…

```
df.head()
```

Out[103…

	target	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 [104…

```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

In [105…

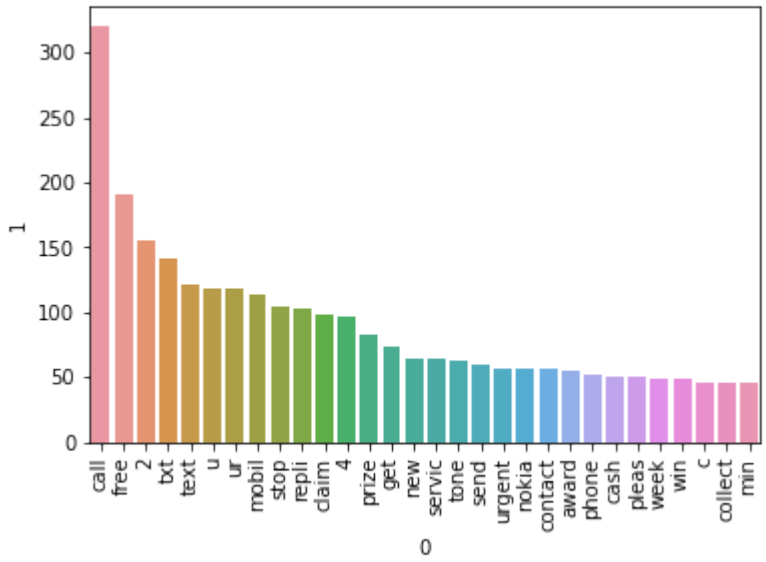
```
len(spam_corpus)
```

Out[105… 9941

In [106…

```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Count
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\Meghna\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [107…

```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [108…

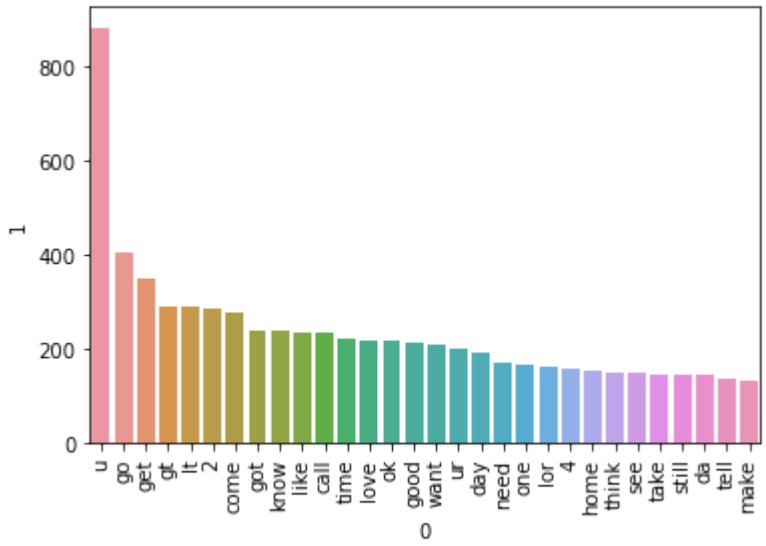
```
len(ham_corpus)
```

Out[108… 35303

In [109…

```
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0],pd.DataFrame(Counte
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\Meghna\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



Model Building

In [110…
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer()
```

In [111…
```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

In [112…
```
X.shape
```

Out[112… (5169, 6677)

In [113…
```
y = df['target'].values
```

In [114…
```
y
```

Out[114… array([0, 0, 1, ..., 0, 0, 0])

In [115…
```
from sklearn.model_selection import train_test_split
```

In [116…
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

Machine Learning (ML) Algorithms

In [117…
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score
```

In [118…
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [119…
```
#GaussianNB
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]]
0.518348623853211
```

In [120…
```
#MultinomialNB
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))
```

```
0.9593810444874274
[[896   0]
 [ 42  96]]
1.0
```

In [121…

```
#BernoulliNB
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))
```

```
0.971953578336557
[[894   2]
 [ 27 111]]
0.9823008849557522
```

In [122…

```
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 [123…

```
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 [124…

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

In [125…

```
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 [126…

```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

Out[126…

```
(0.9729206963249516, 0.9741379310344828)
```

In [128…

```
accuracy_scores = []
precision_scores = []

for name, clf in clfs.items():
    current_accuracy, current_precision = train_classifier(clf, X_train, y_train, X_

    print("For", name)
    print("Accuracy - ", current_accuracy)
    print("Precision - ", current_precision)

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

```
For SVC
Accuracy -  0.9729206963249516
Precision -  0.9741379310344828
For KN
Accuracy -  0.9003868471953579
Precision -  1.0
For NB
Accuracy -  0.9593810444874274
Precision -  1.0
For DT
Accuracy -  0.9342359767891683
Precision -  0.8301886792452831
For LR
Accuracy -  0.9516441005802708
Precision -  0.94
For RF
Accuracy -  0.9700193423597679
Precision -  0.9908256880733946
For AdaBoost
Accuracy -  0.9622823984526112
Precision -  0.9541284403669725
For BgC
Accuracy -  0.9574468085106383
Precision -  0.8615384615384616
For ETC
Accuracy -  0.9777562862669246
Precision -  0.9914529914529915
For GBDT
Accuracy -  0.9516441005802708
Precision -  0.9313725490196079
For xgb
Accuracy -  0.971953578336557
Precision -  0.9504132231404959
```

In [129…

```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'P
```

In [130…

```
performance_df
```

Out[130…

	Algorithm	Accuracy	Precision
1	KN	0.900387	1.000000
2	NB	0.959381	1.000000
8	ETC	0.977756	0.991453
5	RF	0.970019	0.990826
0	SVC	0.972921	0.974138
6	AdaBoost	0.962282	0.954128
10	xgb	0.971954	0.950413
4	LR	0.951644	0.940000
9	GBDT	0.951644	0.931373
7	BgC	0.957447	0.861538
3	DT	0.934236	0.830189

In [131…

```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
```

In [132…

```
performance_df1
```

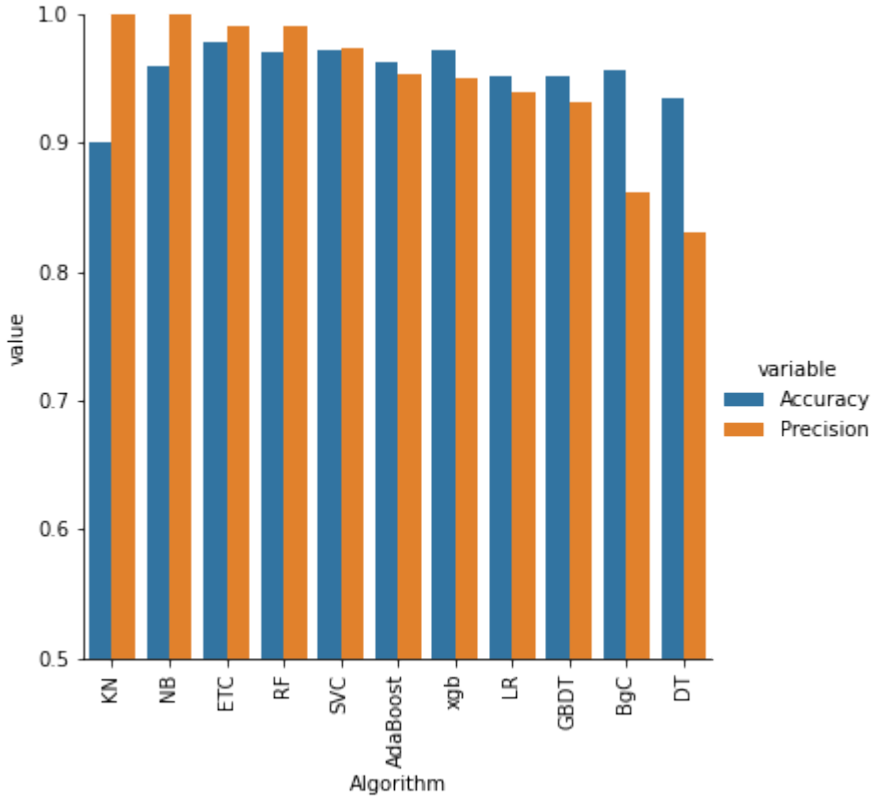
Out[132…

	Algorithm	variable	value
0	KN	Accuracy	0.900387
1	NB	Accuracy	0.959381
2	ETC	Accuracy	0.977756
3	RF	Accuracy	0.970019
4	SVC	Accuracy	0.972921
5	AdaBoost	Accuracy	0.962282
6	xgb	Accuracy	0.971954
7	LR	Accuracy	0.951644
8	GBDT	Accuracy	0.951644
9	BgC	Accuracy	0.957447
10	DT	Accuracy	0.934236
11	KN	Precision	1.000000
12	NB	Precision	1.000000
13	ETC	Precision	0.991453
14	RF	Precision	0.990826
15	SVC	Precision	0.974138
16	AdaBoost	Precision	0.954128
17	xgb	Precision	0.950413
18	LR	Precision	0.940000

	Algorithm	variable	value
19	GBDT	Precision	0.931373
20	BgC	Precision	0.861538
21	DT	Precision	0.830189

In [133…

```
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 [135…

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),
                        'Accuracy_max_ft_3000':accuracy_scores,
                        'Precision_max_ft_3000':precision_scores})
```

In [138…

```
new_df = performance_df.merge(temp_df, on = 'Algorithm')
```

In [139…

```
new_df_scaled = new_df.merge(temp_df,on = 'Algorithm')
```

In [140…

```
new_df_scaled
```

Out[140…

	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000_x	Precision_max_ft_3000_x	Accuracy_max_
0	KN	0.900387	1.000000	0.900387	1.000000	
1	NB	0.959381	1.000000	0.959381	1.000000	
2	ETC	0.977756	0.991453	0.977756	0.991453	

	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000_x	Precision_max_ft_3000_x	Accuracy_max_
3	RF	0.970019	0.990826	0.970019	0.990826	
4	SVC	0.972921	0.974138	0.972921	0.974138	
5	AdaBoost	0.962282	0.954128	0.962282	0.954128	
6	xgb	0.971954	0.950413	0.971954	0.950413	
7	LR	0.951644	0.940000	0.951644	0.940000	
8	GBDT	0.951644	0.931373	0.951644	0.931373	
9	BgC	0.957447	0.861538	0.957447	0.861538	
10	DT	0.934236	0.830189	0.934236	0.830189	

Deep Learning

In [146…

```
%pip install tensorflow
```

```
Collecting tensorflow
  Using cached tensorflow-2.13.1-cp38-cp38-win_amd64.whl (1.9 kB)
  Using cached tensorflow-2.13.0-cp38-cp38-win_amd64.whl (1.9 kB)
Collecting tensorflow-intel==2.13.0
  Using cached tensorflow_intel-2.13.0-cp38-cp38-win_amd64.whl (276.5 MB)
Collecting tensorboard<2.14,>=2.13
  Downloading tensorboard-2.13.0-py3-none-any.whl (5.6 MB)
Requirement already satisfied: setuptools in c:\users\meghna\anaconda3\lib\site-packa
ges (from tensorflow-intel==2.13.0->tensorflow) (52.0.0.post20210125)
Collecting absl-py>=1.0.0
  Downloading absl_py-2.1.0-py3-none-any.whl (133 kB)
Requirement already satisfied: six>=1.12.0 in c:\users\meghna\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.13.0->tensorflow) (1.15.0)
Collecting numpy<=1.24.3,>=1.22
  Downloading numpy-1.24.3-cp38-cp38-win_amd64.whl (14.9 MB)
Collecting gast<=0.4.0,>=0.2.1
  Using cached gast-0.4.0-py3-none-any.whl (9.8 kB)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in c:\users\meghna\ana
conda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorflow) (3.7.4.3)
Collecting grpcio<2.0,>=1.24.3
  Using cached grpcio-1.62.1-cp38-cp38-win_amd64.whl (3.8 MB)
Requirement already satisfied: packaging in c:\users\meghna\anaconda3\lib\site-packag
es (from tensorflow-intel==2.13.0->tensorflow) (20.9)
Collecting keras<2.14,>=2.13.1
  Downloading keras-2.13.1-py3-none-any.whl (1.7 MB)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\meghna\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.13.0->tensorflow) (1.12.1)
Collecting termcolor>=1.1.0
  Using cached termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Requirement already satisfied: h5py>=2.9.0 in c:\users\meghna\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.13.0->tensorflow) (2.10.0)
Collecting libclang>=13.0.0
  Using cached libclang-17.0.6-py2.py3-none-win_amd64.whl (25.8 MB)
Collecting tensorflow-estimator<2.14,>=2.13.0
  Using cached tensorflow_estimator-2.13.0-py2.py3-none-any.whl (440 kB)
Collecting protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=
3.20.3
  Downloading protobuf-4.25.3-cp38-cp38-win_amd64.whl (413 kB)
Collecting tensorflow-io-gcs-filesystem>=0.23.1
  Using cached tensorflow_io_gcs_filesystem-0.31.0-cp38-cp38-win_amd64.whl (1.5 MB)
Collecting astunparse>=1.6.0
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting opt-einsum>=2.3.2
```

```
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Collecting google-pasta>=0.1.1
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Collecting flatbuffers>=23.1.21
  Using cached flatbuffers-24.3.7-py2.py3-none-any.whl (26 kB)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\meghna\anaconda3\lib\si
te-packages (from astunparse>=1.6.0->tensorflow-intel==2.13.0->tensorflow) (0.36.2)
Collecting tensorboard-data-server<0.8.0,>=0.7.0
  Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Collecting google-auth-oauthlib<1.1,>=0.5
  Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\meghna\anaconda3\lib\s
ite-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.
25.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\meghna\anaconda3\lib\site-
packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.0.1)
Collecting markdown>=2.6.8
  Downloading Markdown-3.6-py3-none-any.whl (105 kB)
Collecting google-auth<3,>=1.6.3
  Downloading google_auth-2.28.2-py2.py3-none-any.whl (186 kB)
Collecting pyasn1-modules>=0.2.1
  Downloading pyasn1_modules-0.3.0-py2.py3-none-any.whl (181 kB)
Collecting cachetools<6.0,>=2.0.0
  Downloading cachetools-5.3.3-py3-none-any.whl (9.3 kB)
Collecting rsa<5,>=3.1.4
  Downloading rsa-4.9-py3-none-any.whl (34 kB)
Collecting requests-oauthlib>=0.7.0
  Downloading requests_oauthlib-1.4.0-py2.py3-none-any.whl (24 kB)
Collecting importlib-metadata>=4.4
  Downloading importlib_metadata-7.0.2-py3-none-any.whl (24 kB)
Requirement already satisfied: zipp>=0.5 in c:\users\meghna\anaconda3\lib\site-packag
es (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.14,>=2.13->tensorflo
w-intel==2.13.0->tensorflow) (3.4.1)
Collecting pyasn1<0.6.0,>=0.4.6
  Downloading pyasn1-0.5.1-py2.py3-none-any.whl (84 kB)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\meghna\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==
2.13.0->tensorflow) (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\meghna\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.1
3.0->tensorflow) (2020.12.5)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\meghna\anaconda3\lib\sit
e-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.1
3.0->tensorflow) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\meghna\anaconda3\lib\site-pac
kages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->t
ensorflow) (2.10)
Collecting oauthlib>=3.0.0
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\meghna\anaconda3\lib\site
-packages (from packaging->tensorflow-intel==2.13.0->tensorflow) (2.4.7)
Installing collected packages: pyasn1, rsa, pyasn1-modules, oauthlib, cachetools, req
uests-oauthlib, importlib-metadata, google-auth, tensorboard-data-server, protobuf, n
umpy, markdown, grpcio, google-auth-oauthlib, absl-py, termcolor, tensorflow-io-gcs-f
ilesystem, tensorflow-estimator, tensorboard, opt-einsum, libclang, keras, google-pas
ta, gast, flatbuffers, astunparse, tensorflow-intel, tensorflow
  Attempting uninstall: importlib-metadata
    Found existing installation: importlib-metadata 3.10.0
    Uninstalling importlib-metadata-3.10.0:
      Successfully uninstalled importlib-metadata-3.10.0
  Attempting uninstall: numpy
    Found existing installation: numpy 1.20.1
    Uninstalling numpy-1.20.1:
      Successfully uninstalled numpy-1.20.1
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the package
s that are installed. This behaviour is the source of the following dependency confli
cts.
```

```
scipy 1.6.2 requires numpy<1.23.0,>=1.16.5, but you have numpy 1.24.3 which is incomp
atible.
Successfully installed absl-py-2.1.0 astunparse-1.6.3 cachetools-5.3.3 flatbuffers-2
4.3.7 gast-0.4.0 google-auth-2.28.2 google-auth-oauthlib-1.0.0 google-pasta-0.2.0 grp
cio-1.62.1 importlib-metadata-7.0.2 keras-2.13.1 libclang-17.0.6 markdown-3.6 numpy-
1.24.3 oauthlib-3.2.2 opt-einsum-3.3.0 protobuf-4.25.3 pyasn1-0.5.1 pyasn1-modules-0.
3.0 requests-oauthlib-1.4.0 rsa-4.9 tensorboard-2.13.0 tensorboard-data-server-0.7.2
tensorflow-2.13.0 tensorflow-estimator-2.13.0 tensorflow-intel-2.13.0 tensorflow-io-g
cs-filesystem-0.31.0 termcolor-2.4.0
```

In [147…

```
%pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\meghna\anaconda3\lib\site-packages
(0.11.1)Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: scipy>=1.0 in c:\users\meghna\anaconda3\lib\site-packa
ges (from seaborn) (1.6.2)
Requirement already satisfied: numpy>=1.15 in c:\users\meghna\anaconda3\lib\site-pack
ages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.23 in c:\users\meghna\anaconda3\lib\site-pac
kages (from seaborn) (1.2.4)
Requirement already satisfied: matplotlib>=2.2 in c:\users\meghna\anaconda3\lib\site-
packages (from seaborn) (3.3.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\m
eghna\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: cycler>=0.10 in c:\users\meghna\anaconda3\lib\site-pac
kages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\meghna\anaconda3\lib\site-pa
ckages (from matplotlib>=2.2->seaborn) (10.0.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\meghna\anaconda3\lib
\site-packages (from matplotlib>=2.2->seaborn) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\meghna\anaconda3\lib\sit
e-packages (from matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: six in c:\users\meghna\anaconda3\lib\site-packages (fr
om cycler>=0.10->matplotlib>=2.2->seaborn) (1.15.0)

Requirement already satisfied: pytz>=2017.3 in c:\users\meghna\anaconda3\lib\site-pac
kages (from pandas>=0.23->seaborn) (2021.1)
Collecting numpy>=1.15
  Downloading numpy-1.22.4-cp38-cp38-win_amd64.whl (14.8 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.24.3
    Uninstalling numpy-1.24.3:
      Successfully uninstalled numpy-1.24.3
Successfully installed numpy-1.22.4
```

In [151…

```
pip install tensorflow numpy==1.22.0
```

```
Requirement already satisfied: tensorflow in c:\users\meghna\anaconda3\lib\site-packa
ges (2.13.0)
Collecting numpy==1.22.0
  Downloading numpy-1.22.0-cp38-cp38-win_amd64.whl (14.7 MB)
Requirement already satisfied: tensorflow-intel==2.13.0 in c:\users\meghna\anaconda3
\lib\site-packages (from tensorflow) (2.13.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\meghn
a\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorflow) (0.31.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\meghna\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.13.0->tensorflow) (1.6.3)
Requirement already satisfied: h5py>=2.9.0 in c:\users\meghna\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.13.0->tensorflow) (2.10.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\meghna\anaconda3\lib\site
-packages (from tensorflow-intel==2.13.0->tensorflow) (2.4.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\meghna\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.13.0->tensorflow) (0.4.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\meghna\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.13.0->tensorflow) (1.12.1)
Requirement already satisfied: setuptools in c:\users\meghna\anaconda3\lib\site-packa
```

```
ges (from tensorflow-intel==2.13.0->tensorflow) (52.0.0.post20210125)
Requirement already satisfied: flatbuffers>=23.1.21 in c:\users\meghna\anaconda3\lib
\site-packages (from tensorflow-intel==2.13.0->tensorflow) (24.3.7)
Requirement already satisfied: tensorboard<2.14,>=2.13 in c:\users\meghna\anaconda3\l
ib\site-packages (from tensorflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0 in c:\users\meghna
\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: six>=1.12.0 in c:\users\meghna\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.13.0->tensorflow) (1.15.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\meghna\anaconda3\lib\site-p
ackages (from tensorflow-intel==2.13.0->tensorflow) (2.1.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\meghna\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.13.0->tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\meghna\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.13.0->tensorflow) (1.62.1)
Requirement already satisfied: libclang>=13.0.0 in c:\users\meghna\anaconda3\lib\site
-packages (from tensorflow-intel==2.13.0->tensorflow) (17.0.6)
Requirement already satisfied: packaging in c:\users\meghna\anaconda3\lib\site-packag
es (from tensorflow-intel==2.13.0->tensorflow) (20.9)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in c:\users\meghna\ana
conda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorflow) (3.7.4.3)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\meghna\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.13.0->tensorflow) (3.3.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in c:\users\meghna\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.13.0->tensorflow) (2.13.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!
=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\meghna\anaconda3\lib\site-packages (from tenso
rflow-intel==2.13.0->tensorflow) (4.25.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\meghna\anaconda3\lib\si
te-packages (from astunparse>=1.6.0->tensorflow-intel==2.13.0->tensorflow) (0.36.2)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in c:\users\meghna\anac
onda3\lib\site-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tens
orflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\meghna\anaconda3\lib\site-
packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\meghna\anaconda3\lib
\site-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow)
(2.28.2)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\megh
na\anaconda3\lib\site-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.
0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\meghna\anaconda3\lib\site-
packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.0.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\meghna\anaconda3\lib\s
ite-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.
25.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\meghna\anaconda3\li
b\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-inte
l==2.13.0->tensorflow) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\meghna\anaconda3\lib
\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel
==2.13.0->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\meghna\anaconda3\lib\site-pa
ckages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0
->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\meghna\anaconda3
\lib\site-packages (from google-auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->ten
sorflow-intel==2.13.0->tensorflow) (1.4.0)
Requirement already satisfied: importlib-metadata>=4.4 in c:\users\meghna\anaconda3\l
ib\site-packages (from markdown>=2.6.8->tensorboard<2.14,>=2.13->tensorflow-intel==2.
13.0->tensorflow) (7.0.2)
Requirement already satisfied: zipp>=0.5 in c:\users\meghna\anaconda3\lib\site-packag
es (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.14,>=2.13->tensorflo
w-intel==2.13.0->tensorflow) (3.4.1)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in c:\users\meghna\anaconda3\lib
\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.14,>
=2.13->tensorflow-intel==2.13.0->tensorflow) (0.5.1)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\meghna\anaconda3\lib\sit
e-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.1
```

```
3.0->tensorflow) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\meghna\anaconda3\lib\site-pac
kages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->t
ensorflow) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\meghna\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.1
3.0->tensorflow) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\meghna\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==
2.13.0->tensorflow) (1.26.4)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\meghna\anaconda3\lib\site-
packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard<
2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.2.2)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\meghna\anaconda3\lib\site
-packages (from packaging->tensorflow-intel==2.13.0->tensorflow) (2.4.7)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.22.4
    Uninstalling numpy-1.22.4:
      Successfully uninstalled numpy-1.22.4
Successfully installed numpy-1.22.0
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import tensorflow as tf
```

In [3]:
```
import seaborn as sns
%matplotlib inline
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Flatten, Dens
```

In [6]:
```
data_dl = pd.read_csv('spam.csv')
```

In [7]:
```
data_dl.head(10)
```

Out[7]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN
6	ham	Even my brother is not like to speak with me. ...	NaN	NaN	NaN
7	ham	As per your request 'Melle Melle (Oru Minnamin...	NaN	NaN	NaN
8	spam	WINNER!! As a valued network customer you have...	NaN	NaN	NaN
9	spam	Had your mobile 11 months or more? U R entitle...	NaN	NaN	NaN

In [8]:
```
data_dl.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

In [9]:
```
data_dl.head(10)
```

Out[9]:

	v1	v2
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...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

In [11]:
```
data_dl.rename(columns={'v1': 'label', 'v2': 'message'}, inplace=True)
data_dl.head(10)
```

Out[11]:

	label	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...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

In [12]:
```
# get all the ham and spam
ham_msg = data_dl[data_dl.label =='ham']
spam_msg = data_dl[data_dl.label=='spam']
```

In [13]:
```
ham_msg_text = " ".join(ham_msg.message.to_numpy().tolist())
spam_msg_text = " ".join(spam_msg.message.to_numpy().tolist())
```

In [14]:
```
# downsample the ham msg
ham_msg_df = ham_msg.sample(n = len(spam_msg), random_state = 44)
spam_msg_df = spam_msg
```

In [15]:
```
print(ham_msg_df.shape, spam_msg_df.shape)
```

```
(747, 2) (747, 2)
```

In [16]:
```
# Create a dataframe with these ham and spam msg
msg_df = ham_msg_df.append(spam_msg_df).reset_index(drop=True)
```

In [17]:
```
msg_df.label.value_counts()
```

Out[17]:
```
ham     747
spam    747
Name: label, dtype: int64
```

Next, a final dataframe is created and calculated the text length column. We further explored length of each of the text by lable types. On average, the ham message has length of 73 words whereas spam message has 138.

In [18]:
```
# Get length column for each text
msg_df['text_length'] = msg_df['message'].apply(len)
```

In [20]:
```
#Calculate average length by label types
labels = msg_df.groupby('label').mean()
labels
```

Out[20]:

label	text_length
ham	69.625167
spam	138.429719

Prepare train test data and pre-processing text

First, we need to convert the text label to numeric and split the data into training set and testing set. Also, convert label to numpy arrays to fit deep learning models. 80% of data were used for training and 20% for testing purposes.

In [21]:
```
msg_df['msg_type']= msg_df['label'].map({'ham': 0, 'spam': 1})
msg_label = msg_df['msg_type'].values
```

In [24]:
```
train_msg, test_msg, train_labels, test_labels = train_test_split(msg_df['message'],
```

Tokenization

As deep learning models do not understand text, we need to convert text into numerical representation. For this purpose, a first step is Tokenization. The Tokenizer API from TensorFlow

Keras splits sentences into words and encodes these into integers. Tokenizer will do all required pre-processing such as

- tokenize into word or charachter - here its at word level
- num_words for maximum number of unique tokens hence we can filter out rare words
- filter out punctuation terms
- convert all words to lower case
- convert all words to integer index Below, let's define hyper-parameters used for Tokenization. These hyper-parmaters are briefly discussed as we use these in the code.

In [25]:
```
# hyperparameter

max_len = 50 # pad_sequencs parameter, it idicates we are only going to look for 50
trunc_type = "post" # pad_sequences parameter
padding_type = "post" # pad_sequences parameter
oov_tok = "<OOV>" # out of vocabulary token
vocab_size = 500
```

In [26]:
```
tokenizer = Tokenizer(num_words = vocab_size, char_level=False, oov_token = oov_tok)
tokenizer.fit_on_texts(train_msg)
```

In [27]:
```
word_index = tokenizer.word_index
```

In [28]:
```
# check how many unique tokens have been created
tot_words = len(word_index)
print('unique tokens -> ', tot_words)
```

```
unique tokens ->  4126
```

Sequencing and Padding

Once tokenization is done, let's represent each sentence by sequences of numbers using texts_to_sequences from tokenizer object. Subsequently, we padded the sequence so that we can have same length of each sequence. Sequencing and padding are done for both training and testing data.

In [29]:
```
# sequencing + padding
training_sequences = tokenizer.texts_to_sequences(train_msg)
training_padded = pad_sequences (training_sequences, maxlen = max_len, padding = pad
```

In [30]:
```
testing_sequences = tokenizer.texts_to_sequences(test_msg)
testing_padded = pad_sequences(testing_sequences, maxlen = max_len, padding = paddin
```

- padding = 'pre' or 'post (default pre). By using pre, we'll pad before each sequence and post will pad after each sequence.
- maxlen = maximum length of all sequences. If not provided, by default it will use the maximum length of the longest sentence.
- truncating = 'pre' or 'post' (default 'pre'). If a sequence length is larger than the provided maxlen value then, these values will be truncated to maxlen. 'pre' option will truncate at the

beginning where as 'post' will truncate at the end of the sequences.

In [31]:
```
print('Shape of training tensor: ', training_padded.shape)
print('Shape of testing tensor: ', testing_padded.shape)
```

```
Shape of training tensor:  (1195, 50)
Shape of testing tensor:  (299, 50)
```

In [32]:
```
len(training_sequences[0]), len(training_sequences[1])
```

Out[32]: (27, 24)

In [33]:
```
# padding to same length of 50
len(training_padded[0]), len(training_padded[1])
```

Out[33]: (50, 50)

In [34]:
```
# testing
print(training_padded[0])
```

```
[232   1 396 440   1  23 164   2 119   4   1 155  51   3   1   4   1  54
   6   1   1  36 134   1   1 397   1   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
```

Train the custom model

In [35]:
```
vocab_size = 500 # Number of words you want to tokenize i.e maximum number of words
embeding_dim = 16
```

In [39]:
```
#Dense sentiment model architecture
model = Sequential()
model.add(Embedding(vocab_size, embeding_dim, input_length=max_len))
model.add(GlobalAveragePooling1D())
model.add(Dense(24, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(12, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(6, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [40]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 50, 16)            8000

 global_average_pooling1d (  (None, 16)                0
 GlobalAveragePooling1D)

 dense (Dense)               (None, 24)                408

 dropout (Dropout)           (None, 24)                0

 dense_1 (Dense)             (None, 12)                300
```

```
 dropout_1 (Dropout)         (None, 12)                0

 dense_2 (Dense)             (None, 6)                 78

 dense_3 (Dense)             (None, 1)                 7

=================================================================
Total params: 8793 (34.35 KB)
Trainable params: 8793 (34.35 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Using 'binary_crossentropy' as a loss function because of binary output. We used 'adam' as an optimiser which makes use of momentum to avoid local minima and 'accuracy' as a measure of model performance. We can now use accuracy as the dataset is not imbalanced

In [41]:
```
model.compile(loss='binary_crossentropy',optimizer='adam' ,metrics=['accuracy'])
```

Next let's fit our dense classifier using model.fit argument. It uses padded training data and training labels for training the model and validation data for validating.

- Epoch: Number of times the learning algorithm will work through the entire training data set. We set it to be 20.
- callbacks: callbacks is used to pass the early stopping parameter. EarlyStopping(monitor='val_loss', patience=2) was used to define that we want to monitor the validation loss and if the validation loss is not improved after two epochs, then the model training is stopped. It helps to avoid overfitting problem and indicates when to stop training before the learner begins over-fit.
- verbose =2: lets to print loss and accuracy on each epoch

In [42]:
```
# fitting a dense nerual network model
run = model.fit(training_padded, train_labels,
epochs=30,
validation_data=(testing_padded, test_labels), verbose=2)
```

```
Epoch 1/30
38/38 - 2s - loss: 0.6916 - accuracy: 0.5255 - val_loss: 0.6878 - val_accuracy: 0.819
4 - 2s/epoch - 45ms/step
Epoch 2/30
38/38 - 0s - loss: 0.6799 - accuracy: 0.7682 - val_loss: 0.6607 - val_accuracy: 0.876
3 - 155ms/epoch - 4ms/step
Epoch 3/30
38/38 - 0s - loss: 0.6244 - accuracy: 0.8427 - val_loss: 0.5594 - val_accuracy: 0.889
6 - 158ms/epoch - 4ms/step
Epoch 4/30
38/38 - 0s - loss: 0.4966 - accuracy: 0.8736 - val_loss: 0.3992 - val_accuracy: 0.906
4 - 157ms/epoch - 4ms/step
Epoch 5/30
38/38 - 0s - loss: 0.3604 - accuracy: 0.8929 - val_loss: 0.2688 - val_accuracy: 0.919
7 - 146ms/epoch - 4ms/step
Epoch 6/30
38/38 - 0s - loss: 0.2736 - accuracy: 0.9146 - val_loss: 0.1877 - val_accuracy: 0.943
1 - 150ms/epoch - 4ms/step
Epoch 7/30
38/38 - 0s - loss: 0.2093 - accuracy: 0.9397 - val_loss: 0.1438 - val_accuracy: 0.966
6 - 151ms/epoch - 4ms/step
Epoch 8/30
38/38 - 0s - loss: 0.1770 - accuracy: 0.9473 - val_loss: 0.1294 - val_accuracy: 0.959
9 - 166ms/epoch - 4ms/step
Epoch 9/30
```

```
38/38 - 0s - loss: 0.1495 - accuracy: 0.9515 - val_loss: 0.1087 - val_accuracy: 0.966
6 - 156ms/epoch - 4ms/step
Epoch 10/30
38/38 - 0s - loss: 0.1186 - accuracy: 0.9640 - val_loss: 0.0950 - val_accuracy: 0.969
9 - 162ms/epoch - 4ms/step
Epoch 11/30
38/38 - 0s - loss: 0.1068 - accuracy: 0.9707 - val_loss: 0.0903 - val_accuracy: 0.966
6 - 156ms/epoch - 4ms/step
Epoch 12/30
38/38 - 0s - loss: 0.0980 - accuracy: 0.9707 - val_loss: 0.0902 - val_accuracy: 0.973
2 - 163ms/epoch - 4ms/step
Epoch 13/30
38/38 - 0s - loss: 0.0877 - accuracy: 0.9732 - val_loss: 0.0888 - val_accuracy: 0.969
9 - 154ms/epoch - 4ms/step
Epoch 14/30
38/38 - 0s - loss: 0.0805 - accuracy: 0.9782 - val_loss: 0.0893 - val_accuracy: 0.973
2 - 151ms/epoch - 4ms/step
Epoch 15/30
38/38 - 0s - loss: 0.0693 - accuracy: 0.9808 - val_loss: 0.0911 - val_accuracy: 0.966
6 - 153ms/epoch - 4ms/step
Epoch 16/30
38/38 - 0s - loss: 0.0653 - accuracy: 0.9816 - val_loss: 0.0940 - val_accuracy: 0.966
6 - 166ms/epoch - 4ms/step
Epoch 17/30
38/38 - 0s - loss: 0.0668 - accuracy: 0.9833 - val_loss: 0.0955 - val_accuracy: 0.969
9 - 169ms/epoch - 4ms/step
Epoch 18/30
38/38 - 0s - loss: 0.0550 - accuracy: 0.9841 - val_loss: 0.1111 - val_accuracy: 0.966
6 - 151ms/epoch - 4ms/step
Epoch 19/30
38/38 - 0s - loss: 0.0491 - accuracy: 0.9858 - val_loss: 0.0984 - val_accuracy: 0.966
6 - 152ms/epoch - 4ms/step
Epoch 20/30
38/38 - 0s - loss: 0.0521 - accuracy: 0.9841 - val_loss: 0.1015 - val_accuracy: 0.969
9 - 138ms/epoch - 4ms/step
Epoch 21/30
38/38 - 0s - loss: 0.0430 - accuracy: 0.9916 - val_loss: 0.1029 - val_accuracy: 0.963
2 - 132ms/epoch - 3ms/step
Epoch 22/30
38/38 - 0s - loss: 0.0405 - accuracy: 0.9900 - val_loss: 0.1161 - val_accuracy: 0.966
6 - 142ms/epoch - 4ms/step
Epoch 23/30
38/38 - 0s - loss: 0.0444 - accuracy: 0.9841 - val_loss: 0.1310 - val_accuracy: 0.963
2 - 154ms/epoch - 4ms/step
Epoch 24/30
38/38 - 0s - loss: 0.0298 - accuracy: 0.9933 - val_loss: 0.1186 - val_accuracy: 0.963
2 - 160ms/epoch - 4ms/step
Epoch 25/30
38/38 - 0s - loss: 0.0347 - accuracy: 0.9900 - val_loss: 0.1192 - val_accuracy: 0.956
5 - 133ms/epoch - 3ms/step
Epoch 26/30
38/38 - 0s - loss: 0.0315 - accuracy: 0.9925 - val_loss: 0.1241 - val_accuracy: 0.959
9 - 138ms/epoch - 4ms/step
Epoch 27/30
38/38 - 0s - loss: 0.0296 - accuracy: 0.9908 - val_loss: 0.1251 - val_accuracy: 0.956
5 - 137ms/epoch - 4ms/step
Epoch 28/30
38/38 - 0s - loss: 0.0304 - accuracy: 0.9933 - val_loss: 0.1360 - val_accuracy: 0.956
5 - 134ms/epoch - 4ms/step
Epoch 29/30
38/38 - 0s - loss: 0.0260 - accuracy: 0.9950 - val_loss: 0.1354 - val_accuracy: 0.956
5 - 136ms/epoch - 4ms/step
Epoch 30/30
38/38 - 0s - loss: 0.0216 - accuracy: 0.9941 - val_loss: 0.1422 - val_accuracy: 0.956
5 - 157ms/epoch - 4ms/step
```

In [43]:

```
model.evaluate(testing_padded, test_labels)
```

```
10/10 [==============================] - 0s 3ms/step - loss: 0.1422 - accuracy: 0.956
5
```

Out[43]:
```
[0.14220063388347626, 0.95652174949646]
```

In [44]:
```
metrics = pd.DataFrame(run.history)
metrics[:2]
```

Out[44]:

	loss	accuracy	val_loss	val_accuracy
0	0.691635	0.525523	0.687775	0.819398
1	0.679913	0.768201	0.660657	0.876254

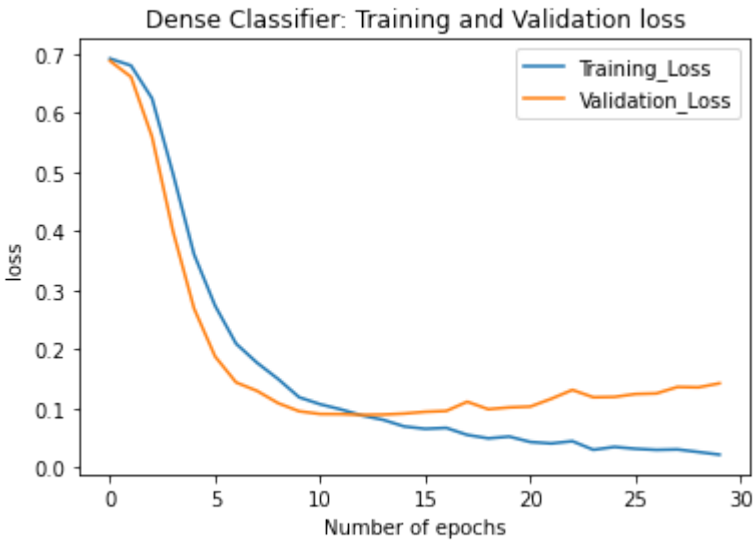
In [45]:
```
# Rename column
metrics.rename(columns = {'loss': 'Training_Loss',
'accuracy': 'Training_Accuracy', 'val_loss': 'Validation_Loss',
'val_accuracy': 'Validation_Accuracy'}, inplace = True)
metrics[:2]
```

Out[45]:

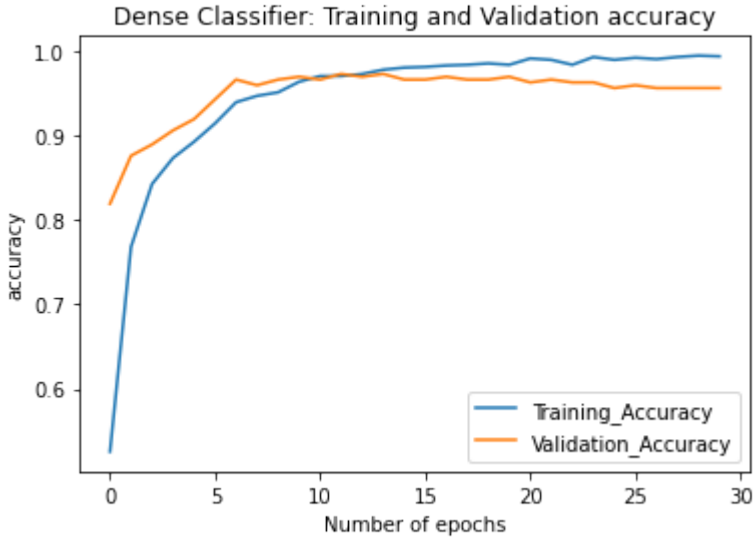
	Training_Loss	Training_Accuracy	Validation_Loss	Validation_Accuracy
0	0.691635	0.525523	0.687775	0.819398
1	0.679913	0.768201	0.660657	0.876254

In [48]:
```
def plot_graphs1(var1, var2, string):
    metrics[[var1, var2]].plot()
    plt.title('Dense Classifier: Training and Validation ' + string)
    plt.xlabel ('Number of epochs')
    plt.ylabel(string)
    plt.legend([var1, var2])
```

In [49]:
```
plot_graphs1('Training_Loss', 'Validation_Loss', 'loss')
```



In [51]:
```
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```



In [52]:
```
#hyperparameters
n_lstm = 20
drop_lstm =0.2
```

In [56]:
```
#LSTM Spam detection architecture
model1 = Sequential()
model1.add(Embedding(vocab_size, embeding_dim, input_length=max_len))
model1.add(LSTM(n_lstm, dropout=drop_lstm, return_sequences=True))
model1.add(LSTM(n_lstm, dropout=drop_lstm, return_sequences=True))
model1.add(Dense(64, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(32, activation='relu'))
model1.add(Dense(16, activation='relu'))
model1.add(Dropout(0.1))
model1.add(Dense(4, activation='relu'))
model1.add(Flatten())
model1.add(Dense(1, activation='sigmoid'))
```

In [57]:
```
model1.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accuracy'
```

In [58]:
```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 50, 16)            8000

 lstm (LSTM)                 (None, 50, 20)            2960

 lstm_1 (LSTM)               (None, 50, 20)            3280

 dense_4 (Dense)             (None, 50, 64)            1344

 dropout_2 (Dropout)         (None, 50, 64)            0

 dense_5 (Dense)             (None, 50, 32)            2080

 dense_6 (Dense)             (None, 50, 16)            528

 dropout_3 (Dropout)         (None, 50, 16)            0

 dense_7 (Dense)             (None, 50, 4)             68
```

```
 flatten (Flatten)           (None, 200)               0

 dense_8 (Dense)             (None, 1)                 201

=================================================================
Total params: 18461 (72.11 KB)
Trainable params: 18461 (72.11 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [59]:
```
model1.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accuracy'
```

In [60]:
```
# Training
num_epochs = 30
history = model1.fit(training_padded, train_labels, epochs=num_epochs, validation_da
```

```
Epoch 1/30
38/38 - 9s - loss: 0.6878 - accuracy: 0.6360 - val_loss: 0.6478 - val_accuracy: 0.779
3 - 9s/epoch - 232ms/step
Epoch 2/30
38/38 - 1s - loss: 0.4106 - accuracy: 0.8360 - val_loss: 0.2119 - val_accuracy: 0.909
7 - 1s/epoch - 39ms/step
Epoch 3/30
38/38 - 2s - loss: 0.1843 - accuracy: 0.9314 - val_loss: 0.1299 - val_accuracy: 0.963
2 - 2s/epoch - 42ms/step
Epoch 4/30
38/38 - 1s - loss: 0.1066 - accuracy: 0.9623 - val_loss: 0.1336 - val_accuracy: 0.963
2 - 1s/epoch - 39ms/step
Epoch 5/30
38/38 - 2s - loss: 0.0900 - accuracy: 0.9715 - val_loss: 0.1176 - val_accuracy: 0.963
2 - 2s/epoch - 40ms/step
Epoch 6/30
38/38 - 1s - loss: 0.0814 - accuracy: 0.9774 - val_loss: 0.1453 - val_accuracy: 0.959
9 - 1s/epoch - 39ms/step
Epoch 7/30
38/38 - 2s - loss: 0.0564 - accuracy: 0.9799 - val_loss: 0.1511 - val_accuracy: 0.949
8 - 2s/epoch - 57ms/step
Epoch 8/30
38/38 - 2s - loss: 0.0370 - accuracy: 0.9883 - val_loss: 0.1703 - val_accuracy: 0.949
8 - 2s/epoch - 47ms/step
Epoch 9/30
38/38 - 2s - loss: 0.0354 - accuracy: 0.9900 - val_loss: 0.1671 - val_accuracy: 0.943
1 - 2s/epoch - 46ms/step
Epoch 10/30
38/38 - 2s - loss: 0.0255 - accuracy: 0.9908 - val_loss: 0.2719 - val_accuracy: 0.939
8 - 2s/epoch - 46ms/step
Epoch 11/30
38/38 - 2s - loss: 0.0466 - accuracy: 0.9849 - val_loss: 0.1735 - val_accuracy: 0.943
1 - 2s/epoch - 44ms/step
Epoch 12/30
38/38 - 2s - loss: 0.0285 - accuracy: 0.9916 - val_loss: 0.1953 - val_accuracy: 0.939
8 - 2s/epoch - 41ms/step
Epoch 13/30
38/38 - 2s - loss: 0.0177 - accuracy: 0.9950 - val_loss: 0.2861 - val_accuracy: 0.946
5 - 2s/epoch - 41ms/step
Epoch 14/30
38/38 - 2s - loss: 0.0114 - accuracy: 0.9983 - val_loss: 0.3147 - val_accuracy: 0.943
1 - 2s/epoch - 41ms/step
Epoch 15/30
38/38 - 2s - loss: 0.0110 - accuracy: 0.9983 - val_loss: 0.3122 - val_accuracy: 0.946
5 - 2s/epoch - 41ms/step
Epoch 16/30
38/38 - 2s - loss: 0.0111 - accuracy: 0.9975 - val_loss: 0.2909 - val_accuracy: 0.933
1 - 2s/epoch - 42ms/step
Epoch 17/30
38/38 - 2s - loss: 0.0352 - accuracy: 0.9866 - val_loss: 0.2893 - val_accuracy: 0.936
```

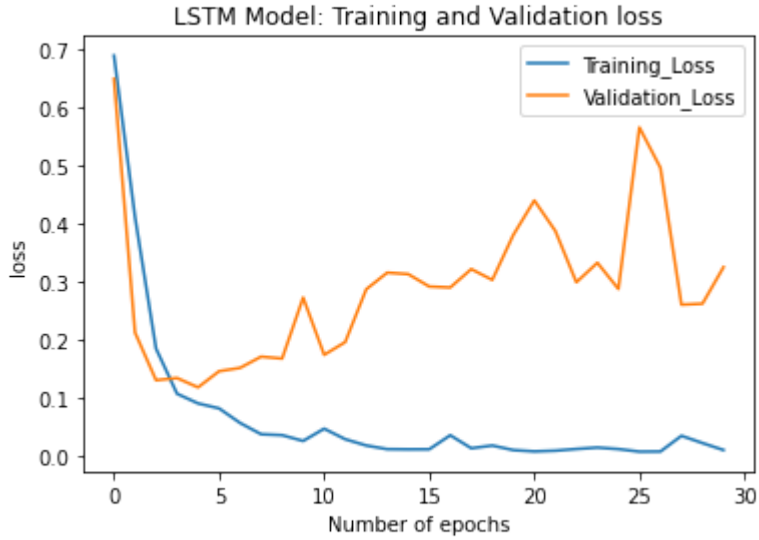
```
5 - 2s/epoch - 42ms/step
Epoch 18/30
38/38 - 2s - loss: 0.0129 - accuracy: 0.9975 - val_loss: 0.3209 - val_accuracy: 0.933
1 - 2s/epoch - 42ms/step
Epoch 19/30
38/38 - 2s - loss: 0.0176 - accuracy: 0.9950 - val_loss: 0.3023 - val_accuracy: 0.936
5 - 2s/epoch - 41ms/step
Epoch 20/30
38/38 - 2s - loss: 0.0098 - accuracy: 0.9992 - val_loss: 0.3796 - val_accuracy: 0.929
8 - 2s/epoch - 40ms/step
Epoch 21/30
38/38 - 2s - loss: 0.0075 - accuracy: 0.9992 - val_loss: 0.4389 - val_accuracy: 0.933
1 - 2s/epoch - 41ms/step
Epoch 22/30
38/38 - 1s - loss: 0.0088 - accuracy: 0.9975 - val_loss: 0.3865 - val_accuracy: 0.936
5 - 1s/epoch - 39ms/step
Epoch 23/30
38/38 - 1s - loss: 0.0117 - accuracy: 0.9967 - val_loss: 0.2982 - val_accuracy: 0.939
8 - 1s/epoch - 39ms/step
Epoch 24/30
38/38 - 1s - loss: 0.0141 - accuracy: 0.9967 - val_loss: 0.3318 - val_accuracy: 0.939
8 - 1s/epoch - 39ms/step
Epoch 25/30
38/38 - 2s - loss: 0.0115 - accuracy: 0.9967 - val_loss: 0.2868 - val_accuracy: 0.939
8 - 2s/epoch - 41ms/step
Epoch 26/30
38/38 - 2s - loss: 0.0071 - accuracy: 0.9992 - val_loss: 0.5643 - val_accuracy: 0.943
1 - 2s/epoch - 40ms/step
Epoch 27/30
38/38 - 2s - loss: 0.0073 - accuracy: 0.9992 - val_loss: 0.4950 - val_accuracy: 0.939
8 - 2s/epoch - 42ms/step
Epoch 28/30
38/38 - 2s - loss: 0.0343 - accuracy: 0.9900 - val_loss: 0.2597 - val_accuracy: 0.929
8 - 2s/epoch - 41ms/step
Epoch 29/30
38/38 - 2s - loss: 0.0219 - accuracy: 0.9925 - val_loss: 0.2614 - val_accuracy: 0.946
5 - 2s/epoch - 42ms/step
Epoch 30/30
38/38 - 2s - loss: 0.0098 - accuracy: 0.9983 - val_loss: 0.3241 - val_accuracy: 0.946
5 - 2s/epoch - 42ms/step
```

In [61]:

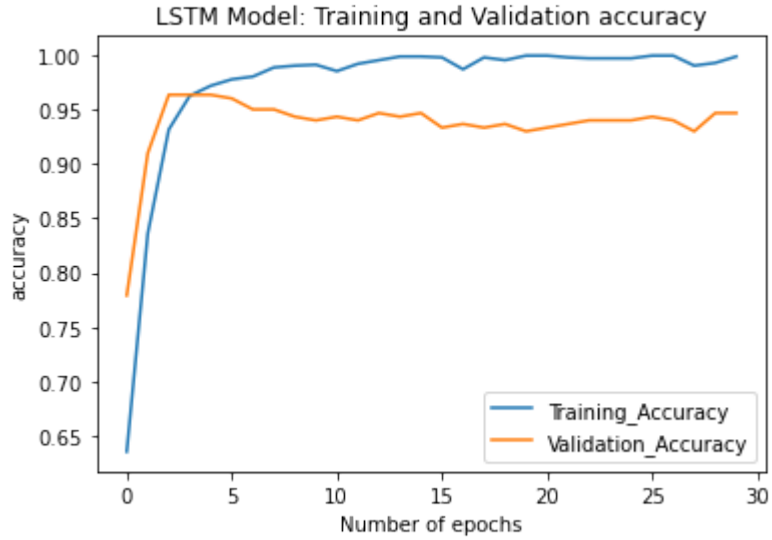
```
metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy',
'val_loss': 'Validation_Loss', 'val_accuracy': 'Validation_Accuracy'}, inplace = Tru
def plot_graphs1(var1, var2, string):
    metrics[[var1, var2]].plot()
    plt.title('LSTM Model: Training and Validation ' + string)
    plt.xlabel ('Number of epochs')
    plt.ylabel(string)
    plt.legend([var1, var2])
```

In [62]:

```
plot_graphs1('Training_Loss', 'Validation_Loss', 'loss')
```



In [63]:
```
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```



Bidirectional LSTM (Bi-LSTM) Spam Detection Architecture

Unlike in LSTM, the Bi-LSTM learns patterns from both before and after a given token within a document. The Bi-LSTM backpropagates in both backward and forward directions in time. Due to this, the computational time is increased compared to LSTM. However, in most of the cases Bi-LSTM results in better accuracy. Below, we can see the Bi-derectional LSTM architechure where only difference than LSTM is that we use Bidirectional wrapper to LSTM.

In [65]:
```
# Biderectional LSTM
model2 = Sequential()
model2.add(Embedding(vocab_size, embeding_dim, input_length=max_len))
model2.add(Bidirectional(LSTM(n_lstm, dropout=drop_lstm, return_sequences=True)))
model2.add(Dense(16, activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(4, activation='relu'))
model2.add(Flatten())
model2.add(Dense(1, activation='sigmoid'))
```

```
model2.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accuracy'
```

In [66]:
```
model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 50, 16)            8000

 bidirectional (Bidirection  (None, 50, 40)            5920
 al)

 dense_9 (Dense)             (None, 50, 16)            656

 dropout_4 (Dropout)         (None, 50, 16)            0

 dense_10 (Dense)            (None, 50, 4)             68

 flatten_1 (Flatten)         (None, 200)               0

 dense_11 (Dense)            (None, 1)                 201

=================================================================
Total params: 14845 (57.99 KB)
Trainable params: 14845 (57.99 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [67]:
```
# Training
num_epochs = 30
history = model2.fit(training_padded, train_labels, epochs=num_epochs, validation_da
```

```
Epoch 1/30
38/38 - 7s - loss: 0.6559 - accuracy: 0.7414 - val_loss: 0.5359 - val_accuracy: 0.846
2 - 7s/epoch - 190ms/step
Epoch 2/30
38/38 - 1s - loss: 0.3714 - accuracy: 0.8661 - val_loss: 0.2126 - val_accuracy: 0.923
1 - 913ms/epoch - 24ms/step
Epoch 3/30
38/38 - 1s - loss: 0.1934 - accuracy: 0.9314 - val_loss: 0.1294 - val_accuracy: 0.959
9 - 964ms/epoch - 25ms/step
Epoch 4/30
38/38 - 1s - loss: 0.1228 - accuracy: 0.9649 - val_loss: 0.1181 - val_accuracy: 0.953
2 - 921ms/epoch - 24ms/step
Epoch 5/30
38/38 - 1s - loss: 0.0837 - accuracy: 0.9741 - val_loss: 0.1009 - val_accuracy: 0.963
2 - 960ms/epoch - 25ms/step
Epoch 6/30
38/38 - 1s - loss: 0.0682 - accuracy: 0.9816 - val_loss: 0.1448 - val_accuracy: 0.959
9 - 956ms/epoch - 25ms/step
Epoch 7/30
38/38 - 1s - loss: 0.0599 - accuracy: 0.9799 - val_loss: 0.1117 - val_accuracy: 0.953
2 - 1s/epoch - 27ms/step
Epoch 8/30
38/38 - 1s - loss: 0.0424 - accuracy: 0.9849 - val_loss: 0.1161 - val_accuracy: 0.959
9 - 966ms/epoch - 25ms/step
Epoch 9/30
38/38 - 1s - loss: 0.0419 - accuracy: 0.9866 - val_loss: 0.1326 - val_accuracy: 0.946
5 - 984ms/epoch - 26ms/step
Epoch 10/30
38/38 - 1s - loss: 0.0239 - accuracy: 0.9925 - val_loss: 0.1368 - val_accuracy: 0.953
2 - 996ms/epoch - 26ms/step
Epoch 11/30
38/38 - 1s - loss: 0.0218 - accuracy: 0.9933 - val_loss: 0.1578 - val_accuracy: 0.956
5 - 982ms/epoch - 26ms/step
```

```
Epoch 12/30
38/38 - 1s - loss: 0.0214 - accuracy: 0.9941 - val_loss: 0.1469 - val_accuracy: 0.949
8 - 1s/epoch - 27ms/step
Epoch 13/30
38/38 - 1s - loss: 0.0146 - accuracy: 0.9967 - val_loss: 0.1505 - val_accuracy: 0.956
5 - 934ms/epoch - 25ms/step
Epoch 14/30
38/38 - 1s - loss: 0.0133 - accuracy: 0.9975 - val_loss: 0.1722 - val_accuracy: 0.946
5 - 916ms/epoch - 24ms/step
Epoch 15/30
38/38 - 1s - loss: 0.0088 - accuracy: 0.9983 - val_loss: 0.1661 - val_accuracy: 0.956
5 - 924ms/epoch - 24ms/step
Epoch 16/30
38/38 - 1s - loss: 0.0078 - accuracy: 0.9992 - val_loss: 0.2007 - val_accuracy: 0.943
1 - 964ms/epoch - 25ms/step
Epoch 17/30
38/38 - 1s - loss: 0.0081 - accuracy: 0.9992 - val_loss: 0.2119 - val_accuracy: 0.939
8 - 959ms/epoch - 25ms/step
Epoch 18/30
38/38 - 1s - loss: 0.0096 - accuracy: 0.9967 - val_loss: 0.2096 - val_accuracy: 0.949
8 - 902ms/epoch - 24ms/step
Epoch 19/30
38/38 - 1s - loss: 0.0090 - accuracy: 0.9992 - val_loss: 0.2201 - val_accuracy: 0.953
2 - 900ms/epoch - 24ms/step
Epoch 20/30
38/38 - 1s - loss: 0.0066 - accuracy: 0.9992 - val_loss: 0.2050 - val_accuracy: 0.946
5 - 907ms/epoch - 24ms/step
Epoch 21/30
38/38 - 1s - loss: 0.0079 - accuracy: 0.9992 - val_loss: 0.2179 - val_accuracy: 0.953
2 - 964ms/epoch - 25ms/step
Epoch 22/30
38/38 - 1s - loss: 0.0057 - accuracy: 0.9992 - val_loss: 0.2138 - val_accuracy: 0.949
8 - 948ms/epoch - 25ms/step
Epoch 23/30
38/38 - 1s - loss: 0.0095 - accuracy: 0.9983 - val_loss: 0.2086 - val_accuracy: 0.956
5 - 942ms/epoch - 25ms/step
Epoch 24/30
38/38 - 1s - loss: 0.0072 - accuracy: 0.9992 - val_loss: 0.1904 - val_accuracy: 0.949
8 - 938ms/epoch - 25ms/step
Epoch 25/30
38/38 - 1s - loss: 0.0074 - accuracy: 0.9992 - val_loss: 0.1868 - val_accuracy: 0.956
5 - 943ms/epoch - 25ms/step
Epoch 26/30
38/38 - 1s - loss: 0.0098 - accuracy: 0.9975 - val_loss: 0.1822 - val_accuracy: 0.956
5 - 963ms/epoch - 25ms/step
Epoch 27/30
38/38 - 1s - loss: 0.0142 - accuracy: 0.9975 - val_loss: 0.2037 - val_accuracy: 0.949
8 - 1s/epoch - 27ms/step
Epoch 28/30
38/38 - 1s - loss: 0.0164 - accuracy: 0.9975 - val_loss: 0.2194 - val_accuracy: 0.946
5 - 936ms/epoch - 25ms/step
Epoch 29/30
38/38 - 1s - loss: 0.0125 - accuracy: 0.9975 - val_loss: 0.1837 - val_accuracy: 0.953
2 - 924ms/epoch - 24ms/step
Epoch 30/30
38/38 - 1s - loss: 0.0088 - accuracy: 0.9983 - val_loss: 0.1992 - val_accuracy: 0.956
5 - 940ms/epoch - 25ms/step
```

In [70]:

```
# Create a dataframe
metrics = pd.DataFrame(history.history)
```

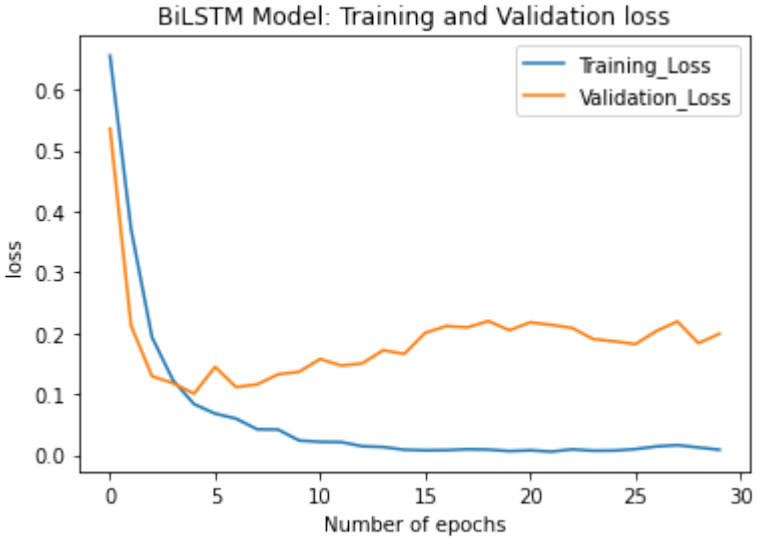
In [72]:

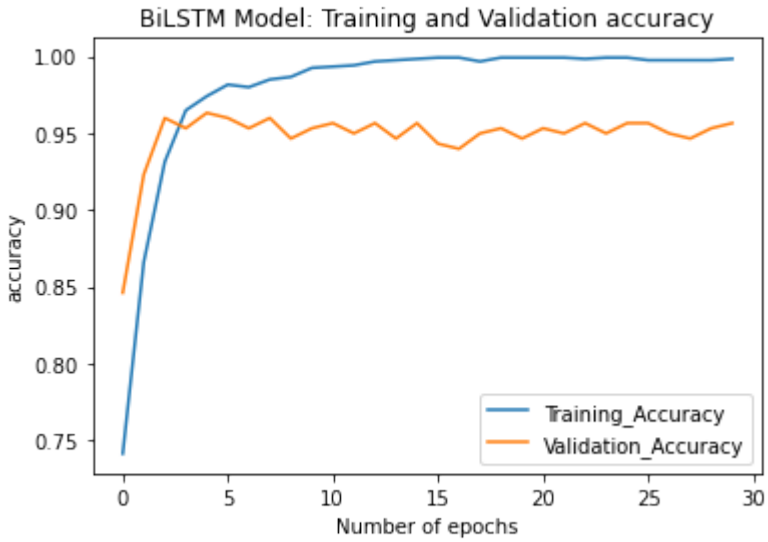
```
# Rename column
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy',
def plot_graphs1(var1, var2, string):
    metrics[[var1, var2]].plot()
    plt.title('BiLSTM Model: Training and Validation ' + string)
    plt.xlabel ('Number of epochs')
```

```
    plt.ylabel(string)
    plt.legend([var1, var2])
```

In [74]:

```
plot_graphs1('Training_Loss', 'Validation_Loss', 'loss')
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```





In [75]:

```
# Comparing the three models used
print(f"Dense architecture loss and accuracy: {model.evaluate(testing_padded, test_l
print(f"LSTM architecture loss and accuracy: {model1.evaluate(testing_padded, test_l
print(f"Bi-LSTM architecture loss and accuracy: {model2.evaluate(testing_padded, tes
```

```
10/10 [==============================] - 0s 2ms/step - loss: 0.1422 - accuracy: 0.956
5
Dense architecture loss and accuracy: [0.14220063388347626, 0.95652174949646]
10/10 [==============================] - 0s 11ms/step - loss: 0.3241 - accuracy: 0.94
65
LSTM architecture loss and accuracy: [0.32410863041877747, 0.9464883208274841]
10/10 [==============================] - 0s 8ms/step - loss: 0.1992 - accuracy: 0.956
5
Bi-LSTM architecture loss and accuracy: [0.19923517107963562, 0.95652174949646]
```

In [80]:
```
def predict_spam(predict_msg, model):
    new_seq = tokenizer.texts_to_sequences(predict_msg)
    padded = pad_sequences(new_seq, maxlen =max_len,
    padding = padding_type, truncating=trunc_type)
    return (model.predict(padded))
```

In [81]:
```
predict_msg = ["You have won $100192",
"where are you?",
"You should click the link below to get flat 100% off on all our goods"]
```

In [82]:
```
predict_spam(predict_msg, model) #custom model
```

```
1/1 [==============================] - 0s 132ms/step
```

Out[82]:
```
array([[0.7758996 ],
       [0.0019241 ],
       [0.89380306]], dtype=float32)
```

In [83]:
```
predict_spam(predict_msg, model1) #lstm
```

```
1/1 [==============================] - 1s 897ms/step
```

Out[83]:
```
array([[9.999994e-01],
       [7.925999e-04],
       [9.949474e-01]], dtype=float32)
```

In [84]:
```
predict_spam(predict_msg, model2) #bi-lstm
```

```
1/1 [==============================] - 1s 775ms/step
```

Out[84]:
```
array([[9.9862516e-01],
       [7.9459260e-04],
       [9.9891853e-01]], dtype=float32)
```

In [ ]:
```

```