In [122…
```
import os
import warnings
warnings.filterwarnings('ignore')
print(os.listdir("C:/Users/anishm/OneDrive - Adobe/Documents/Dataset_csv"))
```

```
['BostonHousing.csv', 'coursera_data.xlsx', 'emails.csv', 'housing.csv', 'melb_data.csv', 'netflix_titles.csv', 'spam.csv']
```

In [123…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [124…
```
data="C:/Users/anishm/OneDrive - Adobe/Documents/Dataset_csv/spam.csv"
df = pd.read_csv(data, encoding='latin-1')[['v1', 'v2']]
df.columns = ['label', 'message']
df.head()
```

Out[124]:

	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...

In [ ]:
```
```

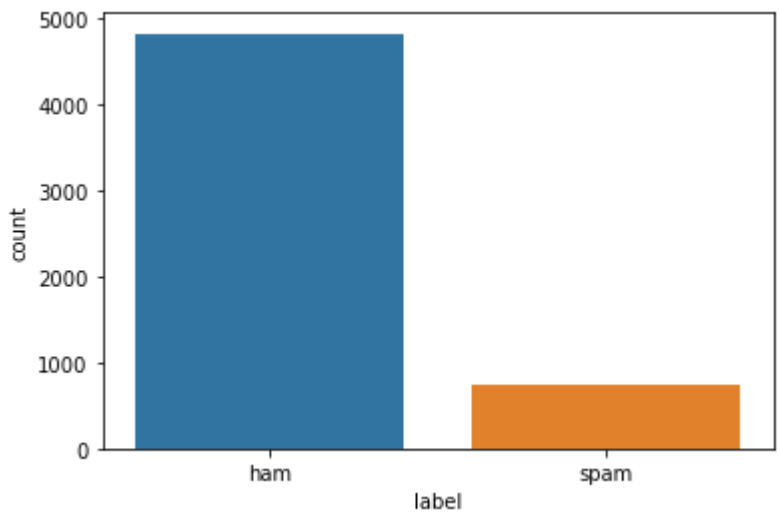
In [125…
```
#Describe dataset and visualize ham/spam count
df.groupby('label').describe()
```

Out[125]:

	message			
	count	unique	top	freq
label				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

In [126…
```
sns.countplot(data=df, x='label')
```

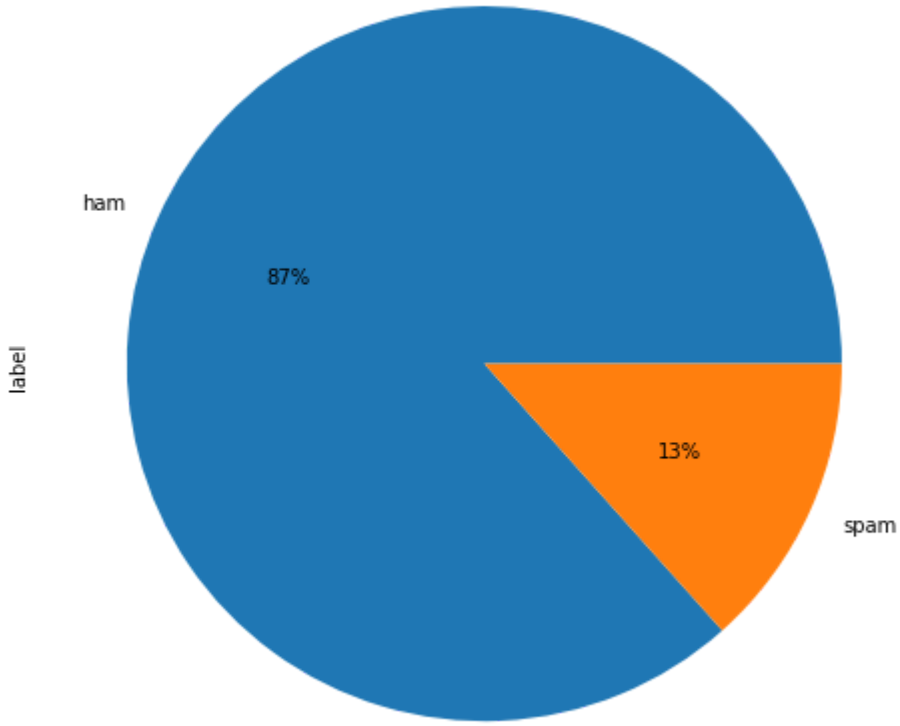
Out[126]:
```
<AxesSubplot:xlabel='label', ylabel='count'>
```



In [127…
```
plt.rcParams["figure.figsize"] = [8,10]
```

```
df.label.value_counts().plot(kind='pie',
autopct='%1.0f%%')
```

Out[127]: <AxesSubplot:ylabel='label'>



In [128…
```
# Our approach:

# Clean and Normalize text
# Convert text into vectors (using bag of words model) that machine learning models c
# Train and test Classifier
# Clean and normalize text
# It will be done in following steps:

# Remove punctuations
# Remove all stopwords
# Apply stemming (converting to normal form of word).
# For example, 'driving car' and 'drives car' becomes drive car
```

In [129…
```
#Cleaning
import string
from nltk.corpus import stopwords
from nltk import PorterStemmer as Stemmer
def process(text):
    # lowercase it
    text = text.lower()
    # remove punctuation
    text = ''.join([t for t in text if t not in string.punctuation])
    # remove stopwords
    text = [t for t in text.split() if t not in stopwords.words('english')]
    # stemming
    st = Stemmer()
    text = [st.stem(t) for t in text]
    # return token list
    return text
```

In [130…
```
#Testing
process('It\'s holiday and Duggu is playing cricket with Param. They are playing very
```

Out[130]: ['holiday', 'duggu', 'play', 'cricket', 'param', 'play', 'well']

In [131…
```
# Test with our dataset
df['message'][:20].apply(process)
```

Out[131]:
```
0     [go, jurong, point, crazi, avail, bugi, n, gre...
1                         [ok, lar, joke, wif, u, oni]
2     [free, entri, 2, wkli, comp, win, fa, cup, fin...
3         [u, dun, say, earli, hor, u, c, alreadi, say]
4     [nah, dont, think, goe, usf, live, around, tho...
5     [freemsg, hey, darl, 3, week, word, back, id, ...
6     [even, brother, like, speak, treat, like, aid,...
7     [per, request, mell, mell, oru, minnaminungint...
8     [winner, valu, network, custom, select, receiv...
9     [mobil, 11, month, u, r, entitl, updat, latest...
10    [im, gonna, home, soon, dont, want, talk, stuf...
11    [six, chanc, win, cash, 100, 20000, pound, txt...
12    [urgent, 1, week, free, membership, å£100000, ...
13    [ive, search, right, word, thank, breather, pr...
14                                    [date, sunday]
15    [xxxmobilemovieclub, use, credit, click, wap, ...
16                                  [oh, kim, watch]
17    [eh, u, rememb, 2, spell, name, ye, v, naughti...
18    [fine, thatåõ, way, u, feel, thatåõ, way, gota...
19    [england, v, macedonia, dont, miss, goalsteam,...
Name: message, dtype: object
```

In [132…
```
# Convert each message to vectors that machine learning models can understand.
# We will do that using bag-of-words model
```

In [133…
```
# We will use TfidfVectorizer. It will convert collection of text documents (SMS corp
# One dimension represent documents and other dimension repesents each unique word in
# If nth term t has occured p times in mth document, (m, n) value in this matrix will

# where [TF-IDF(t)](https://en.wikipedia.org/wiki/Tf–idf) = Term Frequency (TF) * Inv

# Term Frequency (TF) is a measure of how frequent a term occurs in a document.

# TF(t)= Number of times term t appears in document (p) / Total number of terms in th

# Inverse Document Frequency (IDF) is measure of how important term is. For TF, all t
#treated. But, in IDF, for words that occur frequently like 'is' 'the' 'of' are assig
#While terms that occur rarely that can easily help identify class of input features

# Inverse Document Frequency, IDF(t)= loge(Total number of documents / Number of docu

# At end we will have for every message, vectors normalized to unit length equal to s
#(number of unique terms from entire SMS corpus)
```

In [134…
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfv = TfidfVectorizer(analyzer=process)
data = tfidfv.fit_transform(df['message'])
mess = df.iloc[2]['message']
print(mess)
```

```
Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121
to receive entry question(std txt rate)T&C's apply 08452810075over18's
```

In [135…
```
print(tfidfv.transform([mess]))
```

```
  (0, 7741)     0.18906287739887084
  (0, 7708)     0.14471405235314777
  (0, 7276)     0.12336193745345178
  (0, 7099)     0.2190885570936267
  (0, 6959)     0.11759458460817876
  (0, 5856)     0.16027970945850903
  (0, 5815)     0.2330497030932461
  (0, 5768)     0.2330497030932461
  (0, 4592)     0.15903719770411495
```

```
  (0, 3091)     0.11505037200973967
  (0, 2969)     0.16669800498830506
  (0, 2868)     0.4660994061864922
  (0, 2748)     0.3571909758763146
  (0, 2246)     0.20302402339849024
  (0, 2076)     0.19516151371199045
  (0, 1180)     0.16669800498830506
  (0, 833)      0.2190885570936267
  (0, 433)      0.22518719340674634
  (0, 420)      0.22518719340674634
  (0, 413)      0.09987750376879972
  (0, 72)       0.2330497030932461
```

In [136…

```
# A better view
j = tfidfv.transform([mess]).toarray()[0]
print('index\tidf\ttfidf\tterm')
for i in range(len(j)):
    if j[i] != 0:
        print(i, format(tfidfv.idf_[i], '.4f'), format(j[i], '.4f'), tfidfv.get_featu
```

```
index   idf     tfidf   term
72      8.5271  0.2330  08452810075over18
413     3.6544  0.0999  2
420     8.2394  0.2252  2005
433     8.2394  0.2252  21st
833     8.0163  0.2191  87121
1180    6.0993  0.1667  appli
2076    7.1408  0.1952  comp
2246    7.4285  0.2030  cup
2748    6.5346  0.3572  entri
2868    8.5271  0.4661  fa
2969    6.0993  0.1667  final
3091    4.2096  0.1151  free
4592    5.8190  0.1590  may
5768    8.5271  0.2330  questionstd
5815    8.5271  0.2330  ratetc
5856    5.8645  0.1603  receiv
6959    4.3027  0.1176  text
7099    8.0163  0.2191  tkt
7276    4.5137  0.1234  txt
7708    5.2950  0.1447  win
7741    6.9176  0.1891  wkli
```

In [138…

```
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
spam_filter = Pipeline([
    ('vectorizer', TfidfVectorizer(analyzer=process)), # messages to weighted TFIDF s
    ('classifier', MultinomialNB())                    # train on TFIDF vectors with
])

from sklearn.feature_extraction.text import CountVectorizer
vectorizer= CountVectorizer()
message_bow = vectorizer.fit_transform(df['label'])

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df['message'], df['label'], test_
spam_filter.fit(x_train, y_train)
```

Out[138]:

```
Pipeline(steps=[('vectorizer',
                 TfidfVectorizer(analyzer=<function process at 0x000001F3FE4733A0
>)),
                ('classifier', MultinomialNB())])
```

In [139…

```
predictions = spam_filter.predict(x_test)
count = 0
for i in range(len(y_test)):
    if y_test.iloc[i] != predictions[i]:
        count += 1
```

```
print('Total number of test cases', len(y_test))
print('Number of wrong of predictions', count)
```

```
Total number of test cases 1115
Number of wrong of predictions 39
```

In [140…
```
x_test[y_test != predictions]
```

Out[140]:
```
419     Send a logo 2 ur lover - 2 names joined by a h...
3139    sexy sexy cum and text me im wet and warm and ...
3790    Twinks, bears, scallies, skins and jocks are c...
2877    Hey Boys. Want hot XXX pics sent direct 2 ur p...
2377    YES! The only place in town to meet exciting a...
1499    SMS. ac JSco: Energy is high, but u may not kn...
3417    LIFE has never been this much fun and great un...
3358    Sorry I missed your call let's talk when you h...
2412    I don't know u and u don't know me. Send CHAT ...
3862    Oh my god! I've found your number again! I'm s...
659     88800 and 89034 are premium phone services cal...
3109    Good Luck! Draw takes place 28th Feb 06. Good ...
5466    http//tms. widelive.com/index. wml?id=820554ad...
1268    Can U get 2 phone NOW? I wanna chat 2 set up m...
491     Congrats! 1 year special cinema pass for 2 is ...
2246    Hi ya babe x u 4goten bout me?' scammers getti...
2828    Send a logo 2 ur lover - 2 names joined by a h...
3528    Xmas & New Years Eve tickets are now on sale f...
4247    accordingly. I repeat, just text the word ok o...
4142    In The Simpsons Movie released in July 2007 na...
3979                                  ringtoneking 84484
1637    0A$NETWORKS allow companies to bill for SMS, s...
2802                    FreeMsg>FAV XMAS TONES!Reply REAL
3270    You have 1 new voicemail. Please call 08719181...
2294     You have 1 new message. Please call 08718738034.
2619    <Forwarded from 21870000>Hi - this is your Mai...
234     Text & meet someone sexy today. U can find a d...
760     Romantic Paris. 2 nights, 2 flights from å£79 ...
138     You'll not rcv any more msgs from the chat svc...
689     <Forwarded from 448712404000>Please CALL 08712...
879     U have a Secret Admirer who is looking 2 make ...
1216    You have 1 new voicemail. Please call 08719181...
1892    CALL 09090900040 & LISTEN TO EXTREME DIRTY LIV...
2351    Download as many ringtones as u like no restri...
1317    Win the newest  ÛÏHarry Potter and the Order o...
4458    Welcome to UK-mobile-date this msg is FREE giv...
1879    U have a secret admirer who is looking 2 make ...
4309    Someone U know has asked our dating service 2 ...
1673    Monthly password for wap. mobsi.com is 391784....
Name: message, dtype: object
```

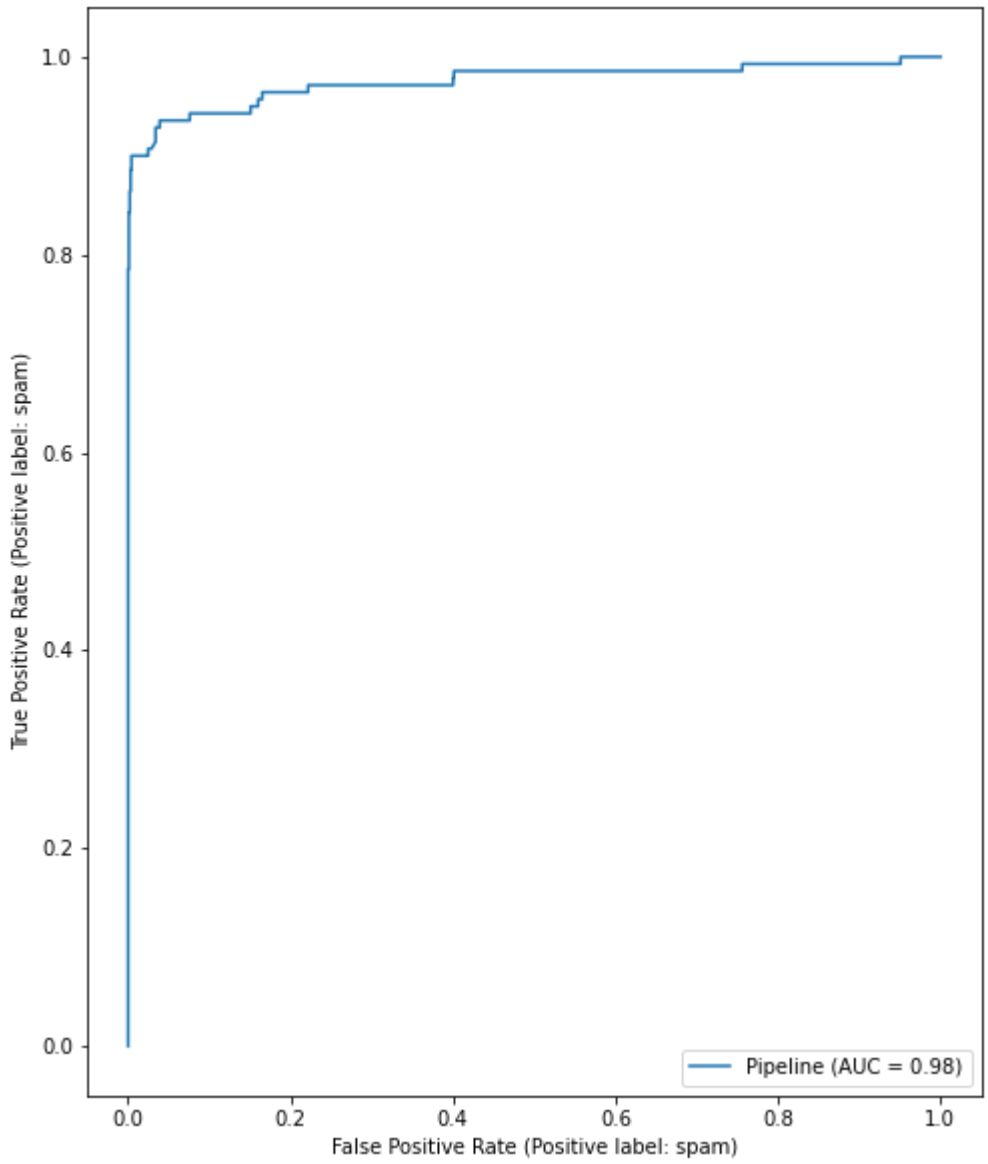
In [141…
```
from sklearn.metrics import classification_report
print(classification_report(predictions, y_test))
```

```
              precision    recall  f1-score   support

         ham       1.00      0.96      0.98      1014
        spam       0.72      1.00      0.84       101

    accuracy                           0.97      1115
   macro avg       0.86      0.98      0.91      1115
weighted avg       0.97      0.97      0.97      1115
```

In [142…
```
# precision column (for ham, it is 1.00), we can say that all number of wrong predict
# (in output of [18]) came from spam predicted as ham. It is ok and cost of predictin
# negligible to that of predicting ham as spam
```

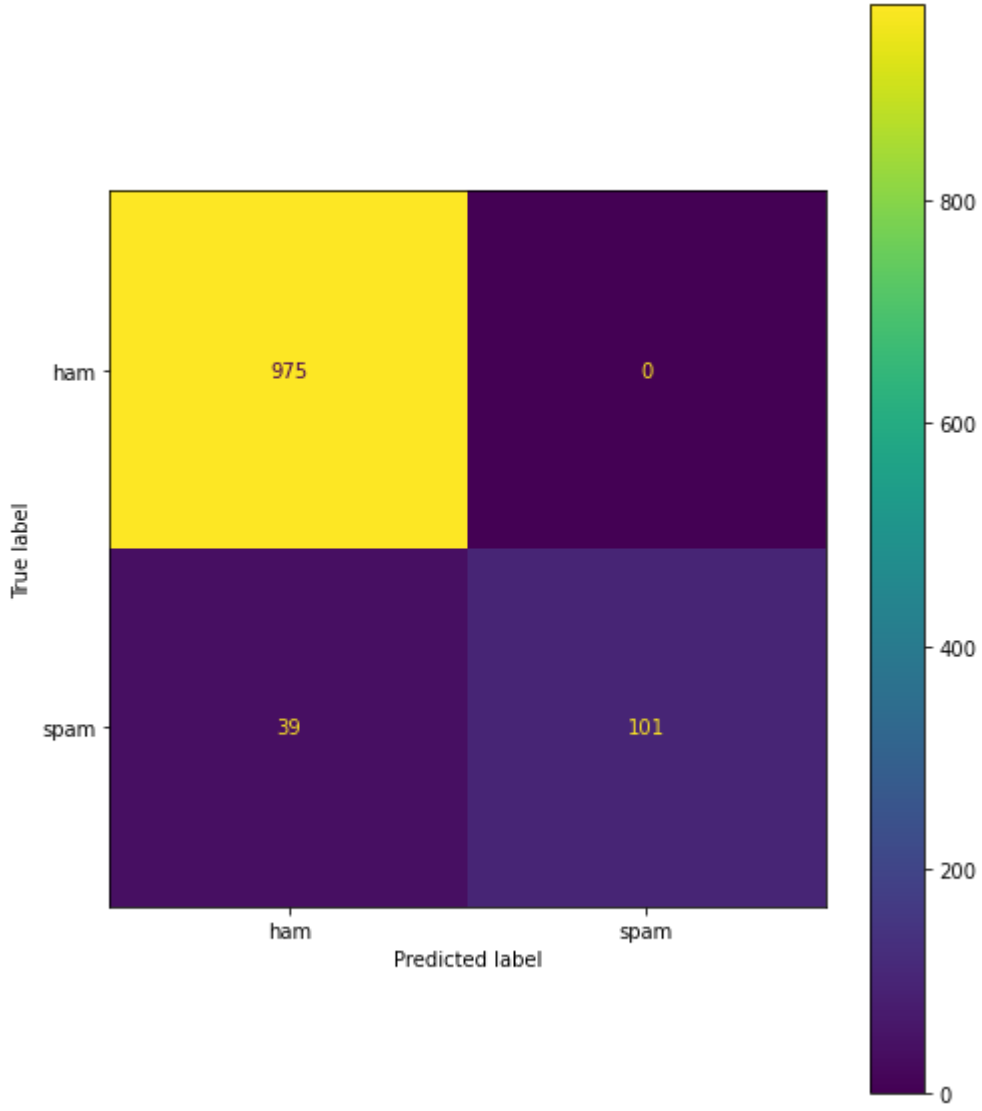
In [143…
```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(spam_filter,x_test,y_test)
```

Out[143]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f3fe45a1f0>



In [182…
```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(spam_filter,x_test,y_test)
```

Out[182]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f3fea0ca00>



In [199…
```
from sklearn.model_selection import KFold, cross_val_score
kfold = KFold(n_splits=5,shuffle=True)
print("Accuracy using Cross Validation is :",np.mean(cross_val_score(spam_filter,y_te
     cv=kfold,scoring="accuracy"))*100," %")
```

```
Accuracy using Cross Validation is : 96.50224215246638  %
```

In [197…
```
def detect_spam(s):
    return spam_filter.predict([s])[0]
detect_spam('Your cash-balance is currently 500 dollar - to maximize your cash-in now
```

Out[197]:
```
'spam'
```

In [198…
```
#The Model is perfectly predicting whether passed message is ham or spam
```

In [ ]:
```
```