In [1]:
```
import pandas as pd
import numpy as np
import tqdm
import nltk
import time
import random
import seaborn as sns
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
from operator import itemgetter
from PIL import Image
```

In [2]:
```
import keras as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve, roc_auc_score, f1_scor
from tensorflow.keras.layers import Embedding, LSTM, Dropout, Dense, Bidirectional
from tensorflow.keras.models import Sequential
from tensorflow.keras.metrics import Recall, Precision
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

%load_ext tensorboard
```

```
WARNING:tensorflow:From C:\Users\ASUS\anaconda3\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse
_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:
```
df = pd.read_csv('messages.csv')
df.head()
df_orig = df.copy()
```

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

Out[4]:
```
(5572, 2)
```

In [5]:
```
pd.set_option('display.max_colwidth', None)

print(df_orig.head(5))
```

```
  Category  \
0      ham
1      ham
2     spam
3      ham
4      ham


Message
0                                               Go until jurong point, crazy.. Available only in bugis n great world l
a e buffet... Cine there got amore wat...
1
Ok lar... Joking wif u oni...
2  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
3                                                                                                             U dun sa
y so early hor... U c already then say...
4                                                                                                 Nah I don't think he
goes to usf, he lives around here though
```

In [6]:
```
df_orig.drop_duplicates(subset=['Message'], keep='first', inplace=True)
```

In [7]:
```
df_orig.shape
```

Out[7]:
```
(5157, 2)
```

In [8]:
```
df_orig.value_counts('Category', normalize=True)
```

Out[8]:
```
Category
ham     0.875703
spam    0.124297
dtype: float64
```

In [9]:
```
df['target'] = [1 if x=='spam' else 0 for  x in df.Category]
df.drop('Category', axis=1, inplace=True)
df.columns = ['message','target']
df.drop_duplicates(subset=['message'], keep='first', inplace=True)
df_spam = df[df.target==1]
df_ham = df[df.target==0]
```

In [10]:
```
df.target.value_counts(normalize=True)
```

Out[10]:
```
0    0.875703
1    0.124297
Name: target, dtype: float64
```

In [11]:
```
X= df.message
y= df.target
```

In [12]:
```
df_orig['word_count'] = df_orig.Message.apply(lambda x: len(x.split(' ')))
```

In [13]:
```
df_orig.groupby('Category')['word_count'].agg(['count','mean','median'])
```

Out[13]:

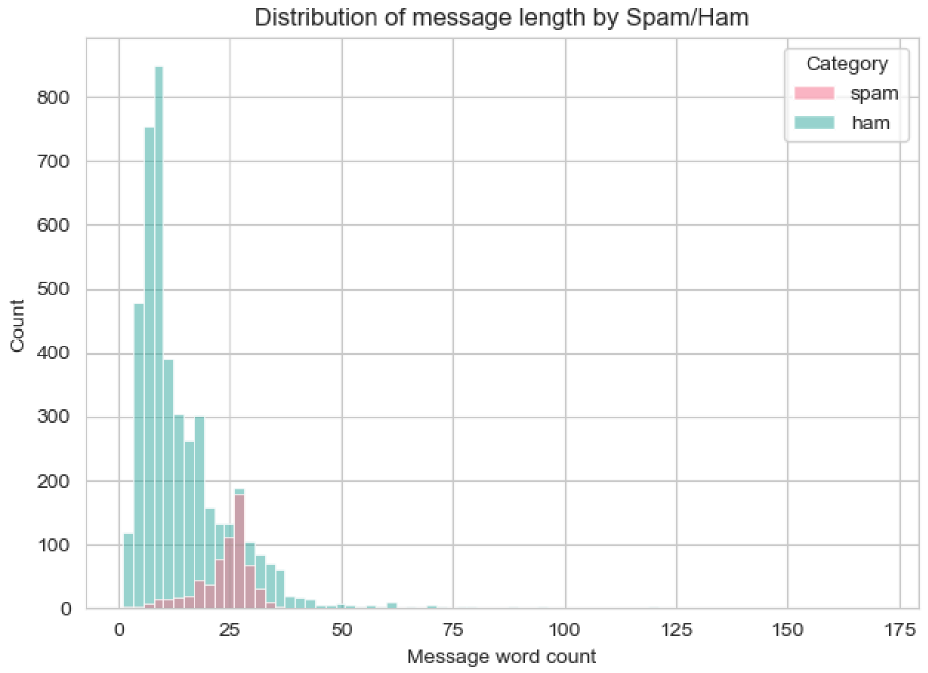
	count	mean	median
Category			
ham	4516	14.328388	11.0
spam	641	23.659906	25.0

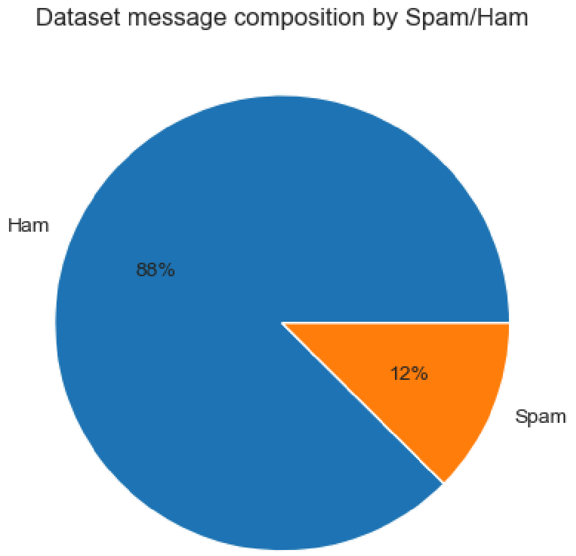
In [14]:
```
df_orig.sample(5)
```

Out[14]:

	Category	Message	word_count
2233	ham	Nothing just getting msgs by dis name wit different no's..	10
174	ham	Bloody hell, cant believe you forgot my surname Mr . Ill give u a clue, its spanish and begins with m...	21
2437	ham	Btw regarding that we should really try to see if anyone else can be our 4th guy before we commit to a random dude	24
1784	ham	No dear i do have free messages without any recharge. Hi hi hi	13
857	ham	Going to take your babe out ?	7

In [15]:
```
# Plot message length by spam/ham

df_orig['Category'] = pd.Categorical(df_orig['Category'], ['spam','ham'])
sns.set_style("whitegrid")

fig, ax = plt.subplots(ncols=2, nrows=1, figsize=(16,5))
sns.histplot(data=df_orig,
             x="word_count",
             hue="Category",
             palette='husl',
             bins=75,
             ax=ax[0],
             legend=True).set_title('Distribution of message length by Spam/Ham')
ax[0].set_xlabel('Message word count')
ax[0].set_ylabel('Count')
colors = sns.color_palette('husl')

ax[1].pie(df_orig.Category.value_counts(normalize=True), labels = ['Ham','Spam'], autopct='%0.f%%')
ax[1].set_title('Dataset message composition by Spam/Ham')
plt.savefig('images/eda.png',bbox_inches='tight',dpi=400, pad_inches=0.1)
plt.show()
```


Distribution of message length by Spam/Ham
Category
spam
ham
Count
800
700
600
500
400
300
200
100
0
0
25
50
75
100
125
150
175
Message word count


Dataset message composition by Spam/Ham
Ham
88%
12%
Spam

```
In [16]: df_orig.sort_values(by='word_count', ascending=False)
```

Out[16]:

	Category	Message	word_count
1085	ham	For me the love should start with attraction.i should feel that I need her every time around me.she should be the first thing which comes in my thoughts.I would start the day and end it with her.she should be there every time I dream.love will be then when my every breath has her name.my life should happen around her.my life will be named to her.I would cry for her.will give all my happiness and take all her sorrows.I will be ready to fight with anyone for her.I will be in love when I will be doing the craziest things for her.love will be when I don't have to proove anyone that my girl is the most beautiful lady on the whole planet.I will always be singing praises for her.love will be when I start up making chicken curry and end up makiing sambar.life will be the most beautiful then.will get every morning and thank god for the day because she is with me.I would like to say a lot..will tell later..	171
1863	ham	The last thing i ever wanted to do was hurt you. And i didn't think it would have. You'd laugh, be embarassed, delete the tag and keep going. But as far as i knew, it wasn't even up. The fact that you even felt like i would do it to hurt you shows you really don't know me at all. It was messy wednesday, but it wasn't bad. The problem i have with it is you HAVE the time to clean it, but you choose not to. You skype, you take pictures, you sleep, you want to go out. I don't mind a few things here and there, but when you don't make the bed, when you throw laundry on top of it, when i can't have a friend in the house because i'm embarassed that there's underwear and bras strewn on the bed, pillows on the floor, that's something else. You used to be good about at least making the bed.	162
2158	ham	Sad story of a Man - Last week was my b'day. My Wife did'nt wish me. My Parents forgot n so did my Kids . I went to work. Even my Colleagues did not wish. As I entered my cabin my PA said, '' Happy B'day Boss !!''. I felt special. She askd me 4 lunch. After lunch she invited me to her apartment. We went there. She said,'' do u mind if I go into the bedroom for a minute ? '' ''OK'', I sed in a sexy mood. She came out 5 minuts latr wid a cake...n My Wife, My Parents, My Kidz, My Friends n My Colleagues. All screaming.. SURPRISE !! and I was waiting on the sofa.. ... . ' NAKED...!	125
1579	ham	How to Make a girl Happy? It's not at all difficult to make girls happy. U only need to be... 1. A friend 2. Companion 3. Lover 4. Chef . . . &lt;#&gt; . Good listener &lt;#&gt; . Organizer &lt;#&gt; . Good boyfriend &lt;#&gt; . Very clean &lt;#&gt; . Sympathetic &lt;#&gt; . Athletic &lt;#&gt; . Warm . . . &lt;#&gt; . Courageous &lt;#&gt; . Determined &lt;#&gt; . True &lt;#&gt; . Dependable &lt;#&gt; . Intelligent . . . &lt;#&gt; . Psychologist &lt;#&gt; . Pest exterminator &lt;#&gt; . Psychiatrist &lt;#&gt; . Healer . . &lt;#&gt; . Stylist &lt;#&gt; . Driver . . Aaniye pudunga venaam..	121
2434	ham	Indians r poor but India is not a poor country. Says one of the swiss bank directors. He says that " &lt;#&gt; lac crore" of Indian money is deposited in swiss banks which can be used for 'taxless' budget for &lt;#&gt; yrs. Can give &lt;#&gt; crore jobs to all Indians. From any village to Delhi 4 lane roads. Forever free power suply to more than &lt;#&gt; social projects. Every citizen can get monthly &lt;#&gt; /- for &lt;#&gt; yrs. No need of World Bank &amp; IMF loan. Think how our money is blocked by rich politicians. We have full rights against corrupt politicians. Itna forward karo ki pura INDIA padhe.g.m."	119
...	...	...	...
284	ham	Okie...	1
2908	ham	ALRITE	1
2508	ham	Yup...	1
783	ham	Beerage?	1
2602	ham	Okie	1

5157 rows × 3 columns

In [17]:
```
# count word appearances
def word_counter(mat):
    count = np.array(mat.sum(axis=0))[0]
    return count

# sort words by frequency
def word_frequency_sorter(words, counts):
    sort_result = sorted(zip(words, list(counts)),
                         key=itemgetter(1), reverse=True)
    return sort_result
```

In [18]:
```
from nltk.corpus import stopwords
stop = stopwords.words('english')

# Series text column
tvec = TfidfVectorizer(stop_words = stop,
                       ngram_range = (1, 2),
                       max_features = 1000,
                       token_pattern = '[A-Za-z0-9]+(?=\\s+)',
                        )

# Fit vectorizer on text data
tvec.fit(df.message)

# Transform data
tvec_mat = tvec.transform(df.message)

# words occuring
words = tvec.get_feature_names_out()

# count word appearances
csum= word_counter(tvec_mat)
```

In [19]:
```
# sort words by frequency
df_word_frequency = pd.DataFrame(word_frequency_sorter(
    words, csum), columns=["word", "frequency_series"])
df_word_frequency.set_index('word', inplace=True)
df_word_frequency.index.name = None
df_word_frequency.head(20)
```

Out[19]:

	frequency_series
u	187.755719
call	109.487576
get	108.333458
2	91.910420
ur	81.290887
got	78.853555
like	78.550733
come	77.469296
go	74.703036
know	68.392965
good	67.684492
going	64.618320
want	61.138508
ok	60.599661
still	59.964543
4	58.781663
send	56.903585
need	55.439239
see	54.492491
text	52.445365

In [20]:
```
# all description text
wrds_cnt = " ".join(wrds for wrds in df.message)
print ("There are {} unique words in all the descriptions.".format(len(wrds_cnt)))

# create stopword list:
stopwords = set(STOPWORDS)

# create word cloud
wc_all = WordCloud(stopwords=stopwords,
                                  max_words=200,
                                  background_color="white")

# generate a word cloud image
wc_all.generate(wrds_cnt)
```

```
There are 413095 unique words in all the descriptions.
```

Out[20]:
```
<wordcloud.wordcloud.WordCloud at 0x29539722200>
```

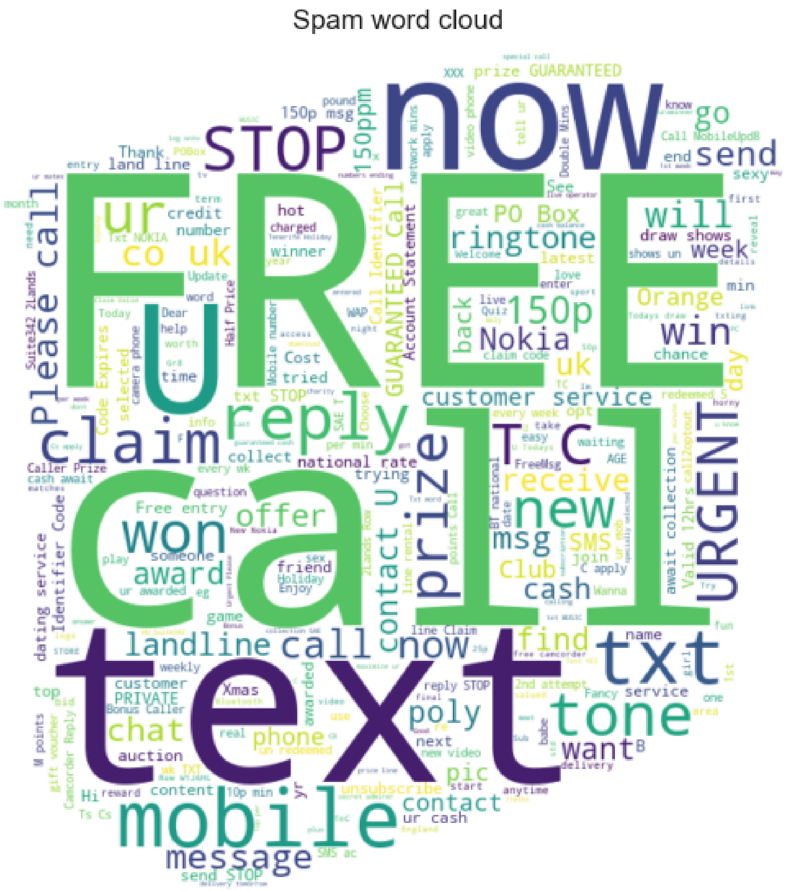
In [21]:
```
# ham description text
wrds_cnt = " ".join(wrds for wrds in df_ham.message)
print ("There are {} unique words in ham the descriptions.".format(len(wrds_cnt)))

# create word mask
mask_ham = np.array(Image.open('C:/Users/ASUS/tensorflow-spam-main/tensorflow-spam-main/images/ham2.png'))

# create word cloud
wc_ham = WordCloud(stopwords=STOPWORDS,
               mask=mask_ham, background_color="white",
               max_words=2000, max_font_size=256,
               random_state=42, width=mask_ham.shape[1],
               height=mask_ham.shape[0])
wc_ham.generate(wrds_cnt);
```

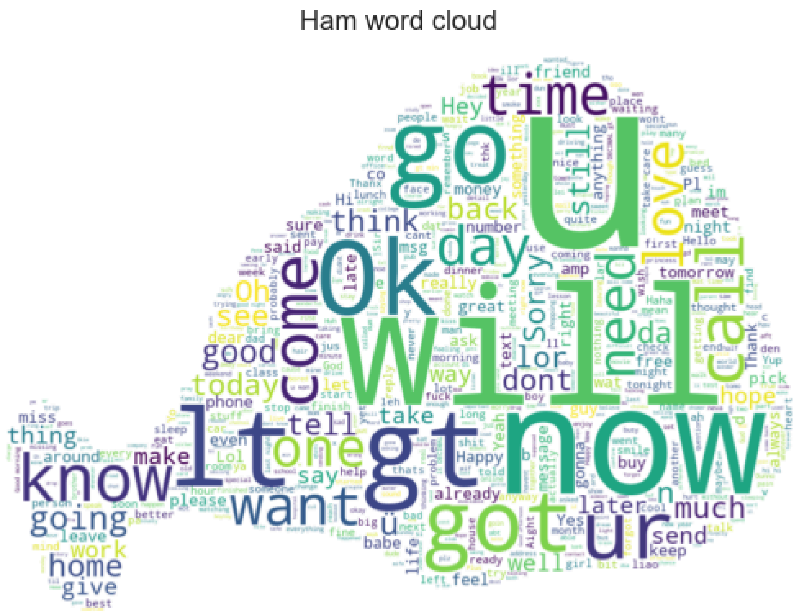
```
There are 324561 unique words in ham the descriptions.
```

In [22]:
```
# spam description text
wrds_cnt = " ".join(wrds for wrds in df_spam.message)
print ("There are {} unique words in spam the descriptions.".format(len(wrds_cnt)))

# create word mask
mask_spam = np.array(Image.open('C:/Users/ASUS/tensorflow-spam-main/tensorflow-spam-main/images/spam2.png'))

# create word cloud
wc_spam = WordCloud(stopwords=STOPWORDS,
               mask=mask_spam, background_color="white",
               max_words=2000, max_font_size=256,
               random_state=42, width=mask_spam.shape[1],
               height=mask_spam.shape[0])
wc_spam.generate(wrds_cnt);
```

```
There are 88533 unique words in spam the descriptions.
```

In [23]:
```
# combine wordcloud plots
sns.set_style("white")
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(14,8))
ax[0].imshow(wc_spam, interpolation="bilinear")
ax[0].set_title('Spam word cloud')
ax[1].imshow(wc_ham, interpolation="bilinear")
ax[1].set_title('Ham word cloud')
fig.set_facecolor("white")
ax[0].axis('off')
ax[1].axis('off')
plt.savefig('C:/Users/ASUS/tensorflow-spam-main/tensorflow-spam-main/images/cloud.png',bbox_inches='tight',dpi=400, pa
plt.show()
```





In [24]:
```
# https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
```

In [25]:
```
SEQUENCE_LENGTH = 50 # the length of all sequences (number of words per sample)
EMBEDDING_SIZE = 100  # Using 100-Dimensional GloVe embedding vectors
TEST_SIZE = 0.25 # ratio of testing set

BATCH_SIZE = 64
EPOCHS = 20 # number of epochs
```

In [26]:
```
# Text tokenization - we have to do this before obtaining the word embeddings
# https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

# vectorizing text, turning each text into sequence of integers.
# Default behaviour is to filter all punctuation, plus tabs and line breaks, minus the ' character.
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X)
# convert to sequence of integers
X = tokenizer.texts_to_sequences(X)

# Save this dictionary for later
word_index = tokenizer.word_index
```

In [27]:
```
print(X[1])
```

```
[49, 304, 1364, 435, 6, 1766]
```

In [28]:
```
# pad sequences at the beginning of each sequence with 0's
# for example if SEQUENCE_LENGTH=4:
# [[5, 3, 2], [5, 1, 2, 3], [3, 4]]
# will be transformed to:
# [[0, 5, 3, 2], [5, 1, 2, 3], [0, 0, 3, 4]]
X = pad_sequences(sequences=X,
                  maxlen=SEQUENCE_LENGTH,
                  padding='pre',
                  truncating='post')
```

In [29]:
```
print(X.shape)
```

```
(5157, 50)
```

In [30]:
```
X
```

Out[30]:
```
array([[   0,    0,    0, ...,   56, 4005,  137],
       [   0,    0,    0, ...,  435,    6, 1766],
       [   0,    0,    0, ...,  707,  437, 4006],
       ...,
       [   0,    0,    0, ...,  100,  240, 9003],
       [   0,    0,    0, ...,  192,   12,   53],
       [   0,    0,    0, ...,    2,   63,  284]])
```

In [31]:
```
y
```

Out[31]:
```
0       0
1       0
2       1
3       0
4       0
       ..
5567    1
5568    0
5569    0
5570    0
5571    0
Name: target, Length: 5157, dtype: int64
```

In [32]:
```
# split and shuffle
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=TEST_SIZE, random_state=5)
# print our data shapes
print("X_train.shape:", X_train.shape)
print("X_test.shape:", X_test.shape)
print("y_train.shape:", y_train.shape)
print("y_test.shape:", y_test.shape)
```

```
X_train.shape: (3867, 50)
X_test.shape: (1290, 50)
y_train.shape: (3867,)
y_test.shape: (1290,)
```

In [33]:
```
# https://machinelearningmastery.com/what-are-word-embeddings/
# https://nlp.stanford.edu/projects/glove/

def get_embedding_vectors(tokenizer, dim=100):
    embedding_index = {}
    with open(f"D:/glove.6B/glove.6B.{dim}d.txt", encoding='utf8') as f:
        for line in tqdm.tqdm(f, "Reading GloVe"):
            values = line.split()
            word = values[0]
            vectors = np.asarray(values[1:], dtype='float32')
            embedding_index[word] = vectors

    word_index = tokenizer.word_index
    embedding_matrix = np.zeros((len(word_index)+1, dim))
    for word, i in word_index.items():
        embedding_vector = embedding_index.get(word)
        if embedding_vector is not None:
            # words not found will be 0s
            embedding_matrix[i] = embedding_vector

    return embedding_matrix
```

In [34]:
```
import tensorflow as tf
tf.compat.v1.executing_eagerly_outside_functions()
```

```
WARNING:tensorflow:From C:\Users\ASUS\AppData\Local\Temp\ipykernel_20772\4192174145.py:2: The name tf.executing_eager
ly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.
```

Out[34]:
```
True
```

In [35]:
```
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR')] # precision-recall curve
```

```
WARNING:tensorflow:From C:\Users\ASUS\anaconda3\lib\site-packages\keras\src\utils\tf_utils.py:585: The name tf.execut
ing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.
```

In [36]:
```
def get_bidirectional_model(tokenizer, lstm_units):
    """
    Constructs the model,
    Embedding vectors => Bi-LSTM => 1 output Fully-Connected neuron with sigmoid activation
    """
    # get the GloVe embedding vectors
    embedding_matrix = get_embedding_vectors(tokenizer)
    # we're going to define the model sequentially https://machinelearningmastery.com/keras-functional-api-deep-learni
    model = Sequential()
    # First, embed the words using loaded GloVe
    model.add(Embedding(len(tokenizer.word_index)+1,
              EMBEDDING_SIZE,
              weights=[embedding_matrix],
              trainable=True,
              mask_zero=True,
              input_length=SEQUENCE_LENGTH))
    # Add bidirectional long short-term memory units
    model.add(Bidirectional(LSTM(lstm_units, recurrent_dropout=0.2)))
    # Add dropout to combat overfitting
    model.add(Dropout(0.2))
    # Add output dense layer with sigmoid for 1/0 classification
    model.add(Dense(1, activation="sigmoid"))
    # compile as rmsprop optimizer
    # aswell as with recall metric
    model.compile(optimizer="rmsprop",
                  loss="binary_crossentropy",
                  metrics=[METRICS])

    model.summary()
    return model
```

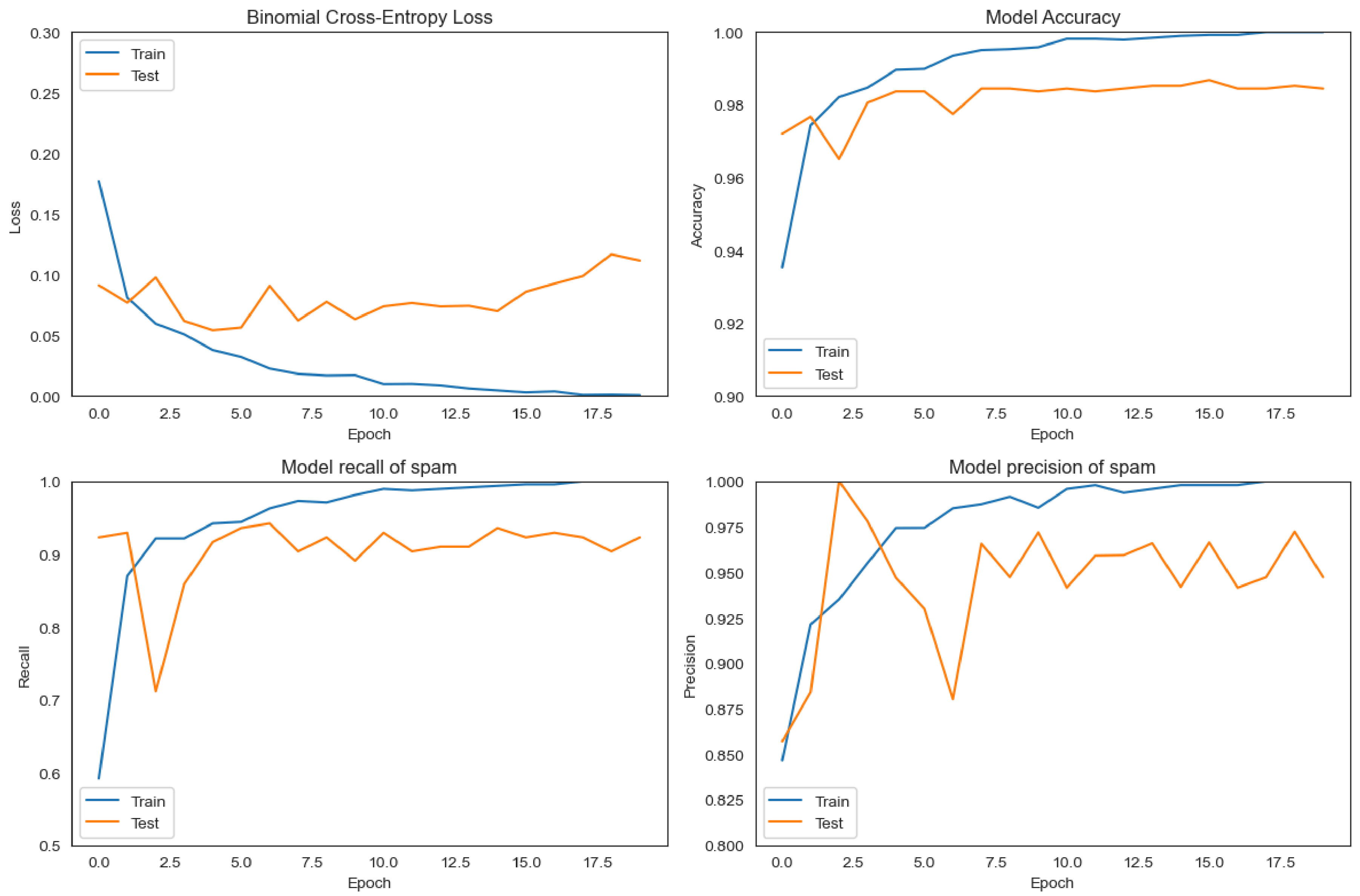
In [37]:
```
# constructs the bidirectional-model with 128 LSTM units
bimodel = get_bidirectional_model(tokenizer=tokenizer, lstm_units=128)
```

```
Reading GloVe: 400000it [00:13, 30618.52it/s]

WARNING:tensorflow:From C:\Users\ASUS\anaconda3\lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.t
rain.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 50, 100)           900500    
                                                                 
 bidirectional (Bidirection  (None, 256)               234496    
 al)                                                             
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 dense (Dense)               (None, 1)                 257       
                                                                 
=================================================================
Total params: 1135253 (4.33 MB)
Trainable params: 1135253 (4.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [38]:
```
# initialize our ModelCheckpoint and TensorBoard callbacks
# model checkpoint for saving best weights
# Change the directory path to a location where you have write permissions
model_checkpoint = ModelCheckpoint("spam_classifier_{val_loss:.2f}.h5", save_best_only=True, verbose=1)
# for better visualization
tensorboard = TensorBoard(log_dir=f"logs/spam_classifier_{time.time()}", histogram_freq=1)
%tensorboard --logdir ./logs

# define model stopping criteria
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=20,
                                             restore_best_weights=True)
# train the model
history = bimodel.fit(X_train, y_train, validation_data=(X_test, y_test),
          batch_size=BATCH_SIZE, #64
          epochs=EPOCHS, #20
          callbacks=[tensorboard, model_checkpoint, early_stop],
          verbose=1)
%tensorboard --logdir logs/fit
```

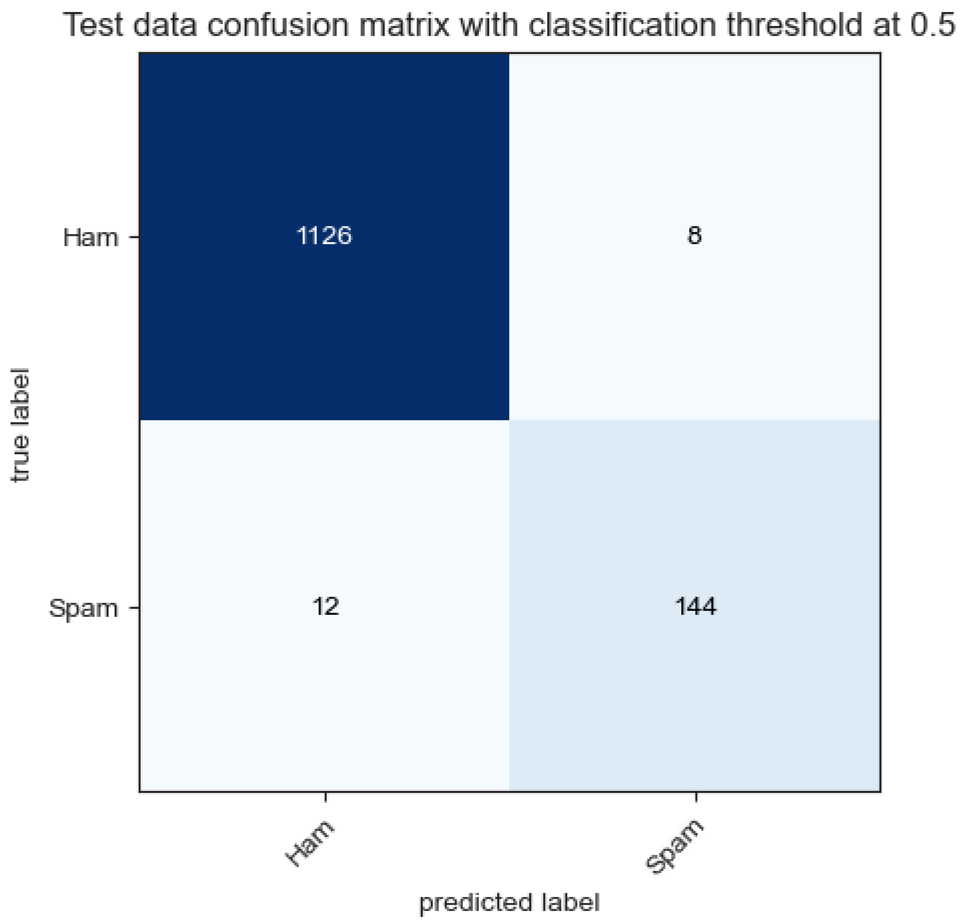
```
Epoch 1/20
WARNING:tensorflow:From C:\Users\ASUS\anaconda3\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragg
ed.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

61/61 [==============================] - ETA: 0s - loss: 0.1770 - tp: 287.0000 - fp: 52.0000 - tn: 3330.0000 - fn:
198.0000 - accuracy: 0.9354 - precision: 0.8466 - recall: 0.5918 - auc: 0.9475 - prc: 0.7950
Epoch 1: val_loss improved from inf to 0.09120, saving model to spam_classifier_0.09.h5
61/61 [==============================] - 19s 190ms/step - loss: 0.1770 - tp: 287.0000 - fp: 52.0000 - tn: 3330.0000
- fn: 198.0000 - accuracy: 0.9354 - precision: 0.8466 - recall: 0.5918 - auc: 0.9475 - prc: 0.7950 - val_loss: 0.09
12 - val_tp: 144.0000 - val_fp: 24.0000 - val_tn: 1110.0000 - val_fn: 12.0000 - val_accuracy: 0.9721 - val_precisio
n: 0.8571 - val_recall: 0.9231 - val_auc: 0.9835 - val_prc: 0.9566
Epoch 2/20
 1/61 [..............................] - ETA: 8s - loss: 0.0810 - tp: 8.0000 - fp: 0.0000e+00 - tn: 55.0000 - fn:
1.0000 - accuracy: 0.9844 - precision: 1.0000 - recall: 0.8889 - auc: 0.9879 - prc: 0.9540

C:\Users\ASUS\anaconda3\lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model
as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Ke
ras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [44]:
```
# Plot classification model performance metrics

fig, ax = plt.subplots(nrows=2,ncols=2, figsize=(12,8))
ax[0,0].plot(history.history['loss'], label='Train')
ax[0,0].plot(history.history['val_loss'], label='Test')
ax[0,0].set_title('Binomial Cross-Entropy Loss')
ax[0,0].set_ylabel('Loss')
ax[0,0].set_ylim(0,0.3)
ax[0,0].set_xlabel('Epoch')
ax[0,0].legend(loc='upper left')
ax[0,1].plot(history.history['accuracy'], label='Train')
ax[0,1].plot(history.history['val_accuracy'], label='Test')
ax[0,1].set_title('Model Accuracy')
ax[0,1].set_ylabel('Accuracy')
ax[0,1].set_ylim(0.9,1)
ax[0,1].set_xlabel('Epoch')
ax[0,1].legend(loc='lower left')
ax[1,0].plot(history.history['recall'], label='Train')
ax[1,0].plot(history.history['val_recall'], label='Test')
ax[1,0].set_title('Model recall of spam')
ax[1,0].set_ylabel('Recall')
ax[1,0].set_ylim(0.5,1)
ax[1,0].set_xlabel('Epoch')
ax[1,0].legend(loc='lower left')
ax[1,1].plot(history.history['precision'], label='Train')
ax[1,1].plot(history.history['val_precision'], label='Test')
ax[1,1].set_title('Model precision of spam')
ax[1,1].set_ylabel('Precision')
ax[1,1].set_ylim(0.8,1)
ax[1,1].set_xlabel('Epoch')
ax[1,1].legend(loc='lower left')
plt.tight_layout()
plt.savefig('images/progression.png',bbox_inches='tight',dpi=400, pad_inches=0.1)
plt.show()
```


Binomial Cross-Entropy Loss
Loss
Epoch
Train
Test
Model Accuracy
Accuracy
Model recall of spam
Recall
Model precision of spam
Precision

In [45]:
```
# Confusion matrix
y_pred = (bimodel.predict(X_test) > 0.5).astype("int32")

plot_confusion_matrix(confusion_matrix(y_test, y_pred), class_names=['Ham','Spam'])
plt.title('Test data confusion matrix with classification threshold at 0.5')
plt.savefig('images/confusion50.png',bbox_inches='tight',dpi=400, pad_inches=0.1)
plt.show()
```

```
41/41 [==============================] - 1s 19ms/step
```



In [46]:
```
# get the loss and metrics
result = bimodel.evaluate(X_test, y_test)
# extract those
loss = result[0]
accuracy = result[5]
precision = result[6]
recall = result[7]
f1 = (2*precision*recall)/(precision+recall)

print(f"Accuracy:    {accuracy*100:.2f}%")
print(f"Precision:   {precision*100:.2f}%")
print(f"Recall:      {recall*100:.2f}%")
print(f"F1 Score:    {f1*100:.2f}%")
```

```
41/41 [==============================] - 1s 34ms/step - loss: 0.1117 - tp: 144.0000 - fp: 8.0000 - tn: 1126.0000 - f
n: 12.0000 - accuracy: 0.9845 - precision: 0.9474 - recall: 0.9231 - auc: 0.9751 - prc: 0.9391
Accuracy:    98.45%
Precision:   94.74%
Recall:      92.31%
F1 Score:    93.51%
```

In [47]:
```
# Sklearn metrics
probs = list(bimodel.predict(X_test))
outcome = [0 if x[0]<0.5 else 1 for x in list(probs)]
dfpred = pd.DataFrame({'true': y_test,'pred':outcome})

print(confusion_matrix(dfpred.true, dfpred.pred))
print(f'Accuracy   {accuracy_score(dfpred.true, dfpred.pred)*100:.2f}%')
print(f'Precision  {precision_score(dfpred.true, dfpred.pred)*100:.2f}%')
print(f'Recall     {recall_score(dfpred.true, dfpred.pred)*100:.2f}%')
print(f'F1-score   {f1_score(dfpred.true, dfpred.pred)*100:.2f}%')
```

```
41/41 [==============================] - 1s 32ms/step
[[1126    8]
 [  12  144]]
Accuracy   98.45%
Precision  94.74%
Recall     92.31%
F1-score   93.51%
```

In [48]:
```
def get_predictions(text):
    sequence = tokenizer.texts_to_sequences([text])
    # pad the sequence
    sequence = pad_sequences(sequence, maxlen=SEQUENCE_LENGTH)
    # get the prediction
    prediction = bimodel.predict(sequence)
    if prediction >0.5:
        return 'Spam'
    else:
        return 'Ham'
```

In [49]:
```
# Spam test
text = "Final Chance! Claim ur £150 worth of discount vouchers today! "
print(get_predictions(text))
```

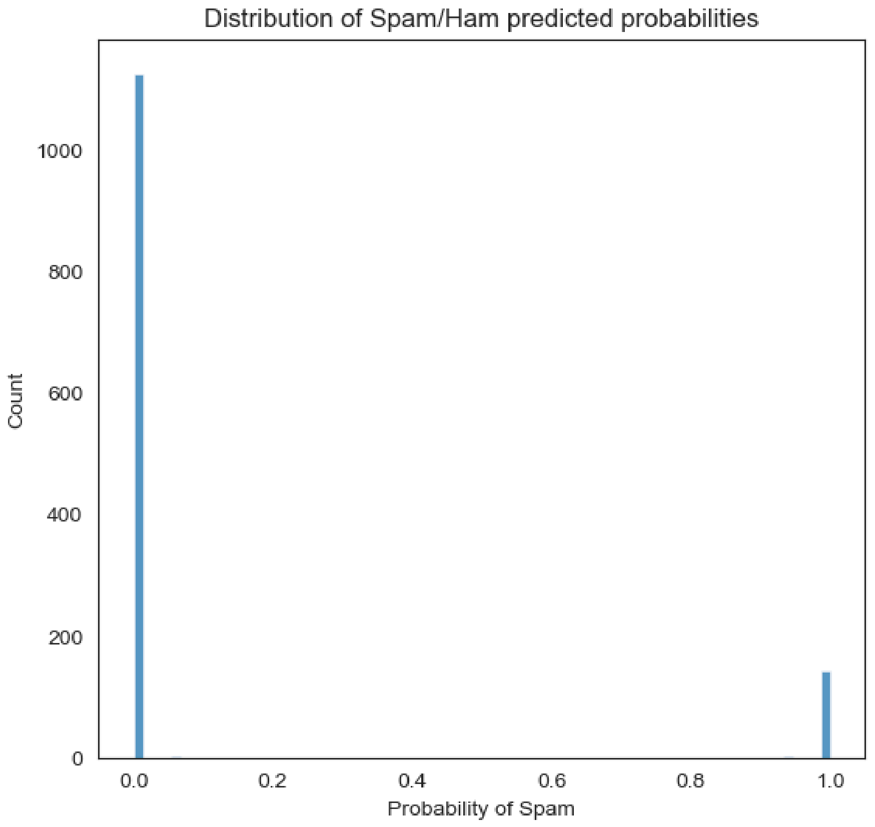
```
1/1 [==============================] - 0s 48ms/step
Spam
```

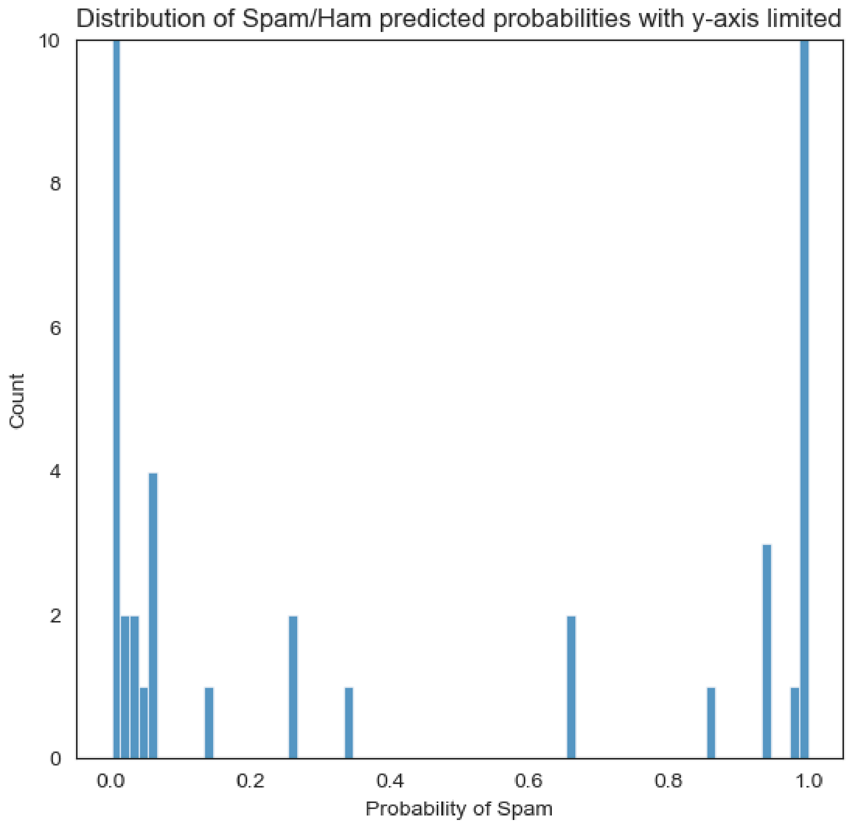
In [50]:
```
# Ham test
text = "Hi man, I was wondering if we can meet tomorrow."
print(get_predictions(text))
```

```
1/1 [==============================] - 0s 52ms/step
Ham
```

In [51]:
```
# Distribution plots of spam probabilities from bimodel

probs_spam = list(bimodel.predict(X_test))
probs_spam

Y_pp = pd.DataFrame(probs_spam, columns=['prob_spam'])
Y_pp['prob_ham']=Y_pp.prob_spam.apply(lambda x: 1-x)


fig, ax = plt.subplots(ncols=2, figsize=(14,6))
sns.histplot(data=Y_pp,
             x="prob_spam",
             palette='husl',
             bins=75,
             ax=ax[1],
             legend=True).set_title('Distribution of Spam/Ham predicted probabilities with y-axis limited')
sns.histplot(data=Y_pp,
             x="prob_spam",
             palette='husl',
             bins=75,
             ax=ax[0],
             legend=True).set_title('Distribution of Spam/Ham predicted probabilities')
ax[1].set_ylim(0,10)
ax[0].set_xlabel('Probability of Spam')
ax[1].set_xlabel('Probability of Spam')
plt.savefig('images/prob_dist.png',bbox_inches='tight',dpi=400, pad_inches=0.1)

plt.show()
```

```
41/41 [==============================] - 1s 33ms/step

C:\Users\ASUS\AppData\Local\Temp\ipykernel_20772\227995102.py:11: UserWarning: Ignoring `palette` because no `hue` va
riable has been assigned.
  sns.histplot(data=Y_pp,
C:\Users\ASUS\AppData\Local\Temp\ipykernel_20772\227995102.py:17: UserWarning: Ignoring `palette` because no `hue` va
riable has been assigned.
  sns.histplot(data=Y_pp,
```


Distribution of Spam/Ham predicted probabilities
Count
Probability of Spam
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0

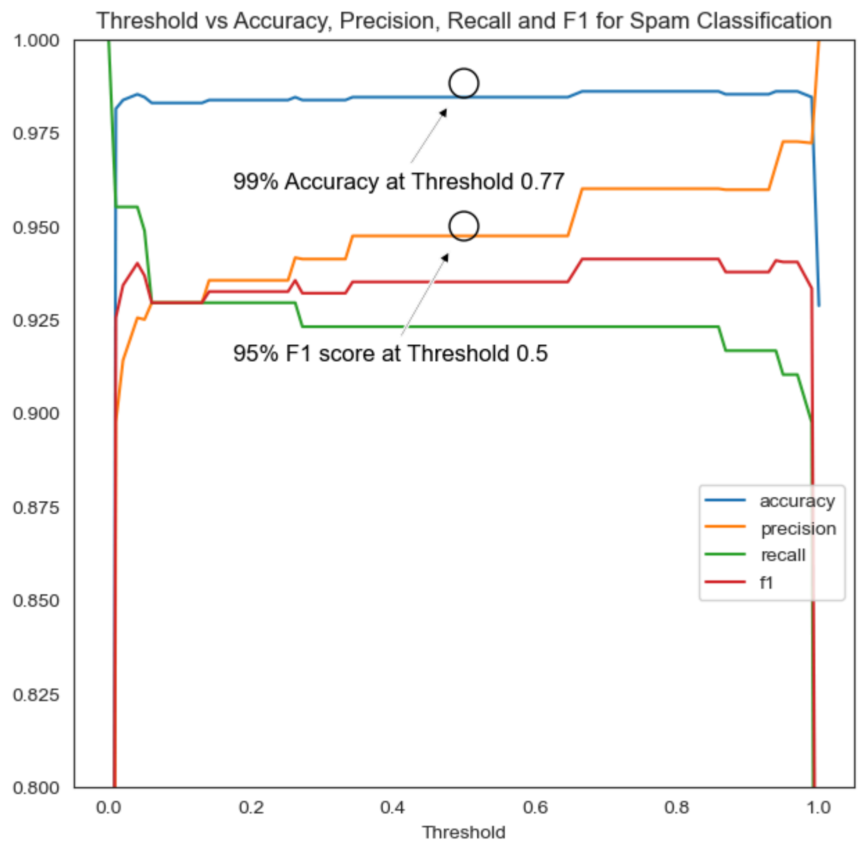

Distribution of Spam/Ham predicted probabilities with y-axis limited
Count
Probability of Spam
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0

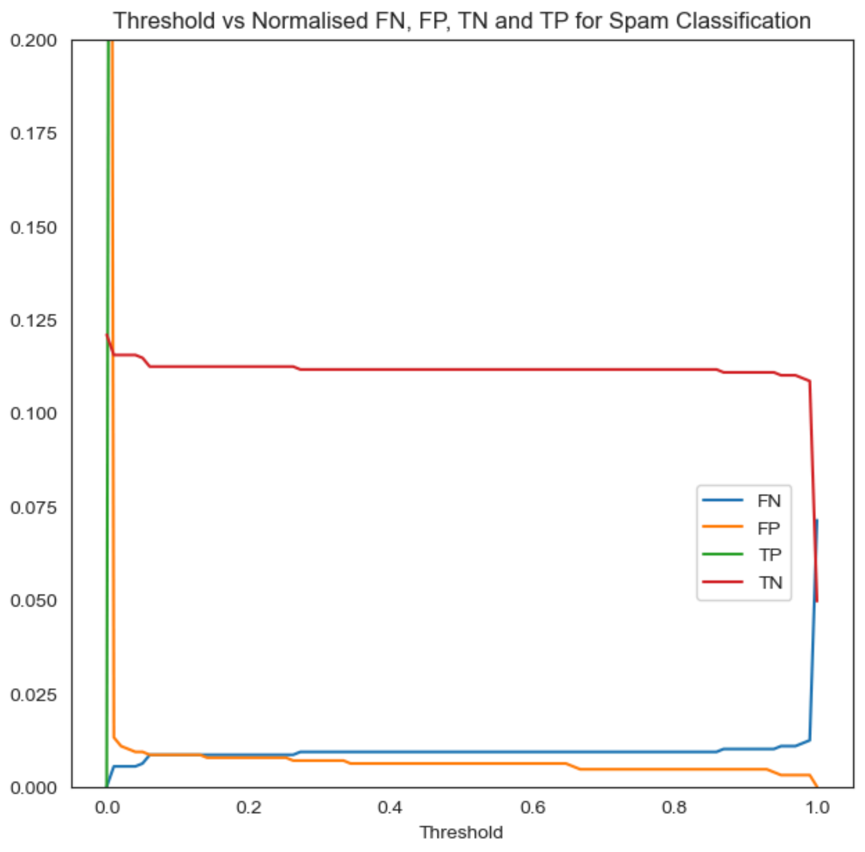
In [52]:
```
Y_pp
```

Out[52]:
```
          prob_spam  prob_ham
0      8.244426e-06  0.999992
1      2.972449e-07  1.000000
2      3.813773e-07  1.000000
3      1.560784e-08  1.000000
4      1.828473e-07  1.000000
...             ...       ...
1285   3.831388e-08  1.000000
1286   1.000000e+00  0.000000
1287   6.806079e-04  0.999319
1288   4.997720e-08  1.000000
1289   2.176357e-08  1.000000

1290 rows × 2 columns
```

In [53]:
```
# Define some functions to help, then plot different scores at different thresholds
def predict_at_threshold(x, threshold=0.5):
    if x >= threshold:
        return 1
    else:
        return 0

def scores_at_thresholds(probs, test_data):
    predictions = [probs.apply(predict_at_threshold, threshold=i) for i in np.linspace(0, 1, 100)]
    scores = np.array(
    [(threshold,
      accuracy_score(test_data, predictions[i]),
      precision_score(test_data, predictions[i],zero_division=1),
      recall_score(test_data, predictions[i]),
      f1_score(test_data, predictions[i]),
      confusion_matrix(test_data,predictions[i], normalize='all')[1,0],
      confusion_matrix(test_data,predictions[i], normalize='all')[0,1],
      confusion_matrix(test_data,predictions[i], normalize='all')[0,0],
      confusion_matrix(test_data,predictions[i], normalize='all')[1,1])

     for i, threshold in enumerate(np.linspace(0, 1, 100))])
    return scores


scores = scores_at_thresholds(Y_pp.prob_spam, y_test);

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(16,7))

# Plot data
ax1.plot(scores[:, 0], scores[:, 1], label='accuracy')
ax1.plot(scores[:, 0], scores[:, 2], label='precision')
ax1.plot(scores[:, 0], scores[:, 3], label='recall')
ax1.plot(scores[:, 0], scores[:, 4], label='f1')
ax2.plot(scores[:, 0], scores[:, 5], label='FN')
ax2.plot(scores[:, 0], scores[:, 6], label='FP')
ax2.plot(scores[:, 0], scores[:, 7], label='TP')
ax2.plot(scores[:, 0], scores[:, 8], label='TN')

# Adjust legends, titles and labels
ax1.legend(loc=[0.8, 0.25])
ax1.set_xlabel('Threshold')
```

```
ax1.set_title('Threshold vs Accuracy, Precision, Recall and F1 for Spam Classification')
ax1.set_ylim(0.8,1)
ax2.legend(loc=[0.8, 0.25])
ax2.set_xlabel('Threshold')
ax2.set_title('Threshold vs Normalised FN, FP, TN and TP for Spam Classification')
ax2.set_ylim(0.0,0.2)

# Annotations
circle_rad = 15  # This is the radius, in points
point=[0.5,0.9885]
ax1.plot(point[0], point[1], 'o', ms=circle_rad * 1, mec='black', mfc='none', mew=1)
ax1.annotate('99% Accuracy at Threshold 0.77', xy=point, xytext=(-120, -55),
            textcoords='offset points',
            color='black', size='large',
            arrowprops=dict(
                arrowstyle='simple,tail_width=0.1,head_width=0.5,head_length=0.5',
                facecolor='black', shrinkB=circle_rad * 1))

point=[0.5,0.95]
ax1.plot(point[0], point[1], 'o', ms=circle_rad * 1, mec='black', mfc='none', mew=1)
ax1.annotate('95% F1 score at Threshold 0.5', xy=point, xytext=(-120, -70),
            textcoords='offset points',
            color='black', size='large',
            arrowprops=dict(
                arrowstyle='simple,tail_width=0.1,head_width=0.5,head_length=0.5',
                facecolor='black', shrinkB=circle_rad * 1))

# point=[0.78,0.025]
# ax2.plot(point[0], point[1], 'o', ms=circle_rad * 1, mec='black', mfc='none', mew=1)
# ax2.annotate('FP=2.5% at Threshold 0.78', xy=point, xytext=(-90, 60),
#             textcoords='offset points',
#             color='black', size='large',
#             arrowprops=dict(
#                 arrowstyle='simple,tail_width=0.1,head_width=0.5,head_length=0.5',
#                 facecolor='black', shrinkB=circle_rad * 1))
plt.savefig('images/threshold.png',bbox_inches='tight',dpi=400, pad_inches=0.1)
plt.show();
```





In [54]:
```
Y_pp['pred_class_thresh'] = Y_pp.prob_spam.apply(predict_at_threshold,
                                                 threshold=0.5)
print(classification_report(y_test, Y_pp.pred_class_thresh))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1134
           1       0.95      0.92      0.94       156

    accuracy                           0.98      1290
   macro avg       0.97      0.96      0.96      1290
weighted avg       0.98      0.98      0.98      1290
```

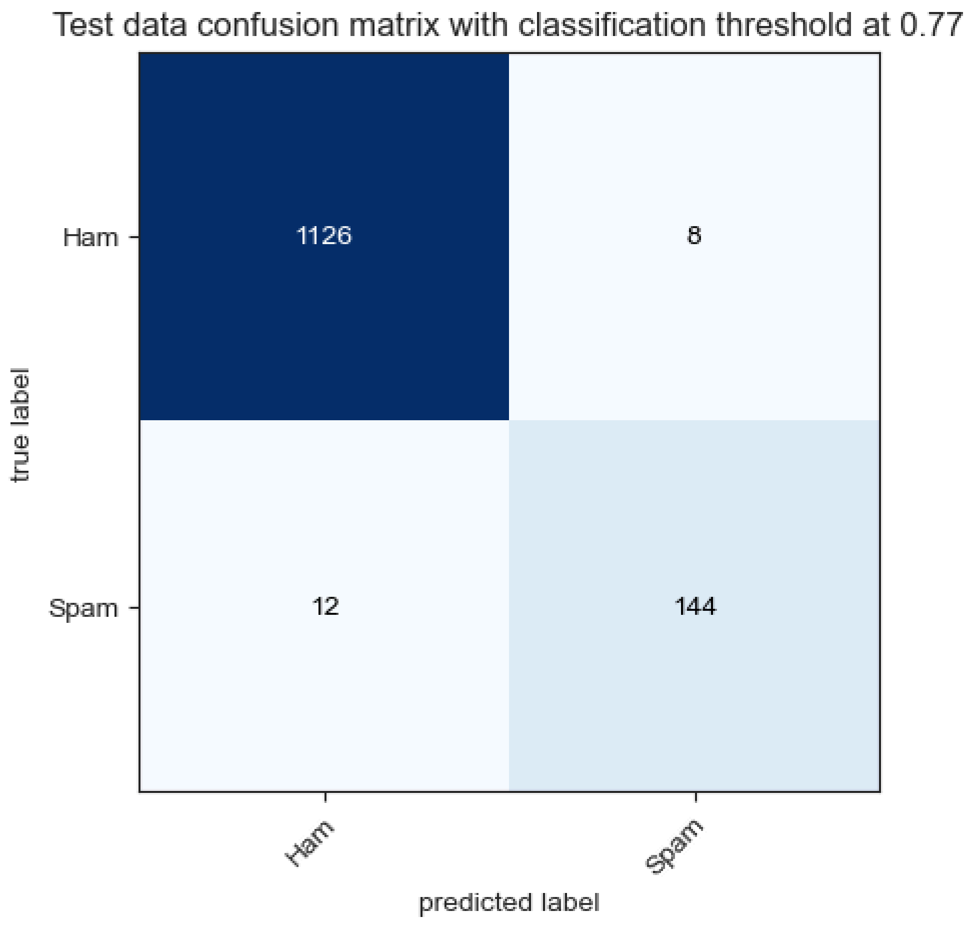
In [55]:

```
Y_pp['pred_class_thresh'] = Y_pp.prob_spam.apply(predict_at_threshold,
                                                 threshold=0.77)
print(classification_report(y_test, Y_pp.pred_class_thresh))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1134
           1       0.96      0.92      0.94       156

    accuracy                           0.99      1290
   macro avg       0.97      0.96      0.97      1290
weighted avg       0.99      0.99      0.99      1290
```

```
In [56]: # Confusion matrix
y_pred = (bimodel.predict(X_test) > 0.5).astype("int32")

plot_confusion_matrix(confusion_matrix(y_test, y_pred), class_names=['Ham','Spam'])
plt.title('Test data confusion matrix with classification threshold at 0.77')
plt.savefig('images/confusion77.png',bbox_inches='tight',dpi=400, pad_inches=0.1)
plt.show()
```

```
41/41 [==============================] - 1s 31ms/step
```



In [57]:
```
# Sklearn metrics
probs = list(bimodel.predict(X_test))
outcome = [0 if x[0]<0.77 else 1 for x in list(probs)]
dfpred = pd.DataFrame({'true': y_test,'pred':outcome})

print(confusion_matrix(dfpred.true, dfpred.pred))
print(f'Accuracy   {accuracy_score(dfpred.true, dfpred.pred)*100:.2f}%')
print(f'Precision  {precision_score(dfpred.true, dfpred.pred)*100:.2f}%')
print(f'Recall     {recall_score(dfpred.true, dfpred.pred)*100:.2f}%')
print(f'F1-score   {f1_score(dfpred.true, dfpred.pred)*100:.2f}%')
```

```
41/41 [==============================] - 1s 32ms/step
[[1128    6]
 [  12  144]]
Accuracy   98.60%
Precision  96.00%
Recall     92.31%
F1-score   94.12%
```

In [58]:
```
text = "Hello bro."
print(get_predictions(text))
```

```
1/1 [==============================] - 0s 48ms/step
Ham
```

In [59]:
```
text = "Flat 50% off ,shop ,shop,shop."
print(get_predictions(text))
```

```
1/1 [==============================] - 0s 52ms/step
Ham
```

In [60]:
```
text = "Thanks for your subscription to Ringtone UK your mobile will be charged £5/month Please confirm by replying YE
print(get_predictions(text))
```

```
1/1 [==============================] - 0s 44ms/step
Spam
```

In [61]:
```
text = "Thanks for your subscription to Ringtone UK your mobile will be charged £5/month Please confirm by replying YE
print(get_predictions(text))
```

```
1/1 [==============================] - 0s 48ms/step
Spam
```

In [ ]: