Assignment Date	15 October 2022
Student Name	MANIMARAN A
Student Roll Number	E1195028
Maximum Marks	2marks
Team ID	PNT2022TMID45335
Project	AI BASED DISCOURSE FOR BAKING INDUSTRY

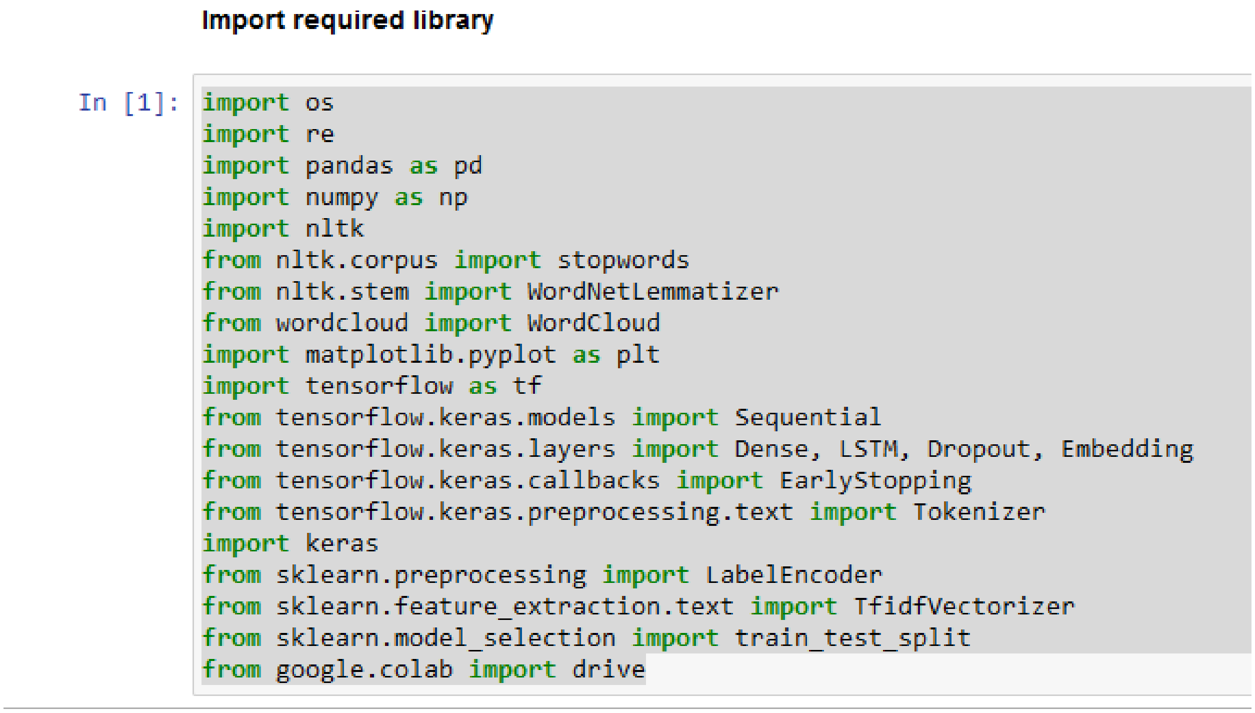
ASSIGNMENT-4

SMS SPAM CLASSIFICATION

Import required library

Solution:

```
import os
import re
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Embedding
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.text import Tokenizer
import keras
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from google.colab import drive
```



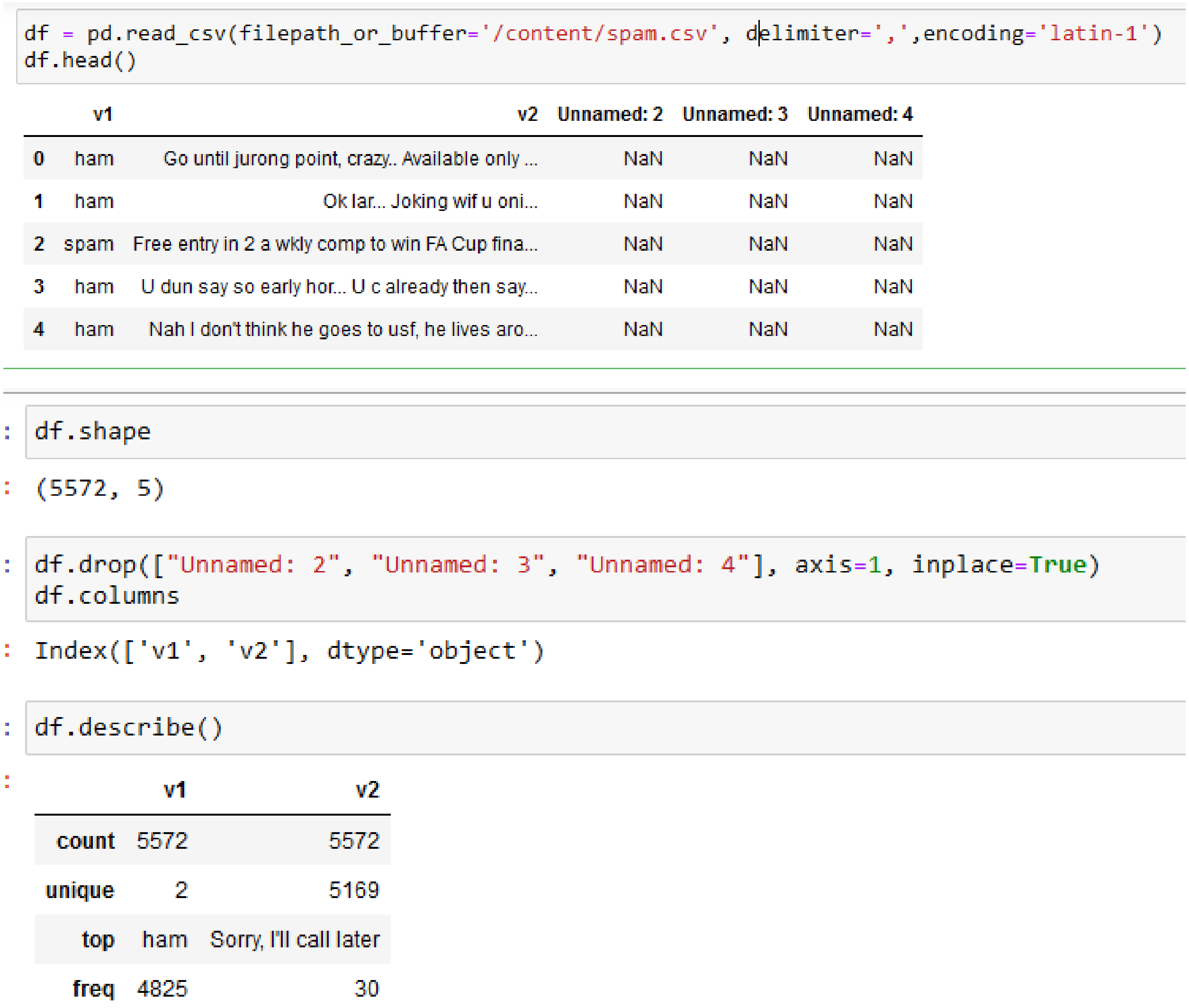
Import required library

```
In [1]: import os
        import re
        import pandas as pd
        import numpy as np
        import nltk
        from nltk.corpus import stopwords
        from nltk.stem import WordNetLemmatizer
        from wordcloud import WordCloud
        import matplotlib.pyplot as plt
        import tensorflow as tf
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, LSTM, Dropout, Embedding
        from tensorflow.keras.callbacks import EarlyStopping
        from tensorflow.keras.preprocessing.text import Tokenizer
        import keras
        from sklearn.preprocessing import LabelEncoder
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from google.colab import drive
```

Read dataset

Solution:

```
df = pd.read_csv(filepath_or_buffer='/content/spam.csv', delimiter=',',encoding='latin-1')
df.head()

df.shape

df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1, inplace=True)
df.columns

df.describe()
df.isna().sum()
df.duplicated().sum()
df = df.drop_duplicates()
df.duplicated().sum()
df['v1'].hist(bins=3)
```



```
df = pd.read_csv(filepath_or_buffer='/content/spam.csv', delimiter=',',encoding='latin-1')
df.head()
```

	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

```
df.shape
```

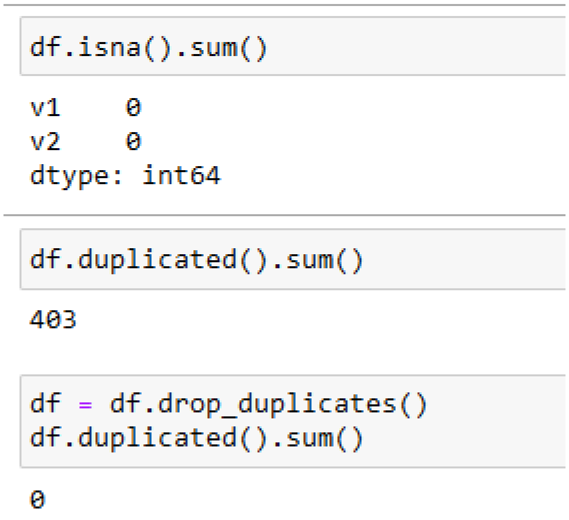
```
(5572, 5)
```

```
df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1, inplace=True)
df.columns
```

```
Index(['v1', 'v2'], dtype='object')
```

```
df.describe()
```

	v1	v2
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30



```
df.isna().sum()
```

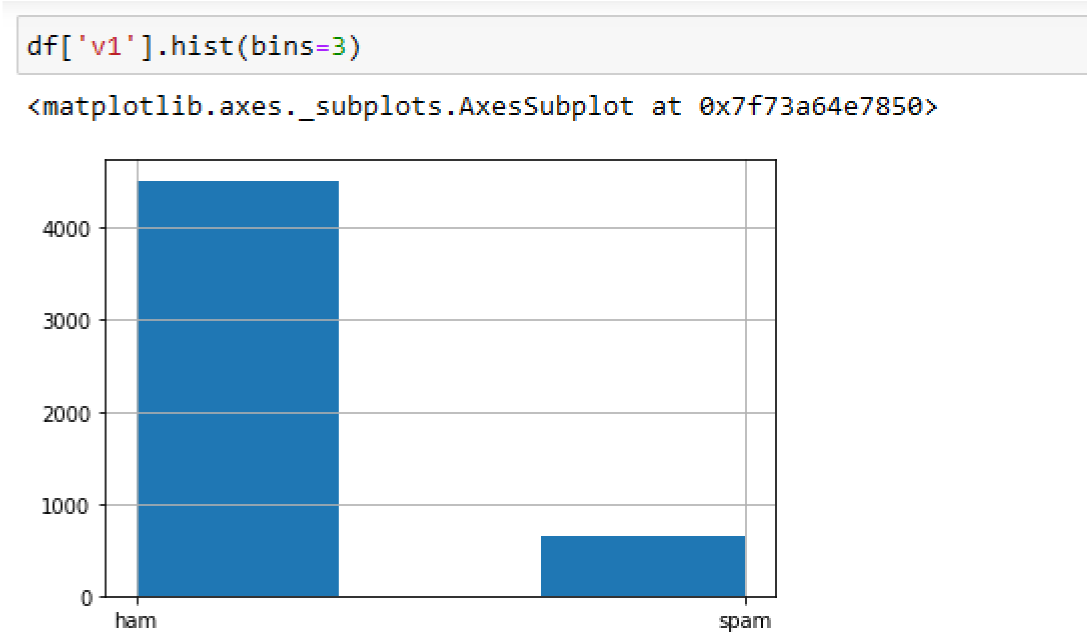
```
v1    0
v2    0
dtype: int64
```

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

```
403
```

```
df = df.drop_duplicates()
df.duplicated().sum()
```

```
0
```



```
df['v1'].hist(bins=3)
```

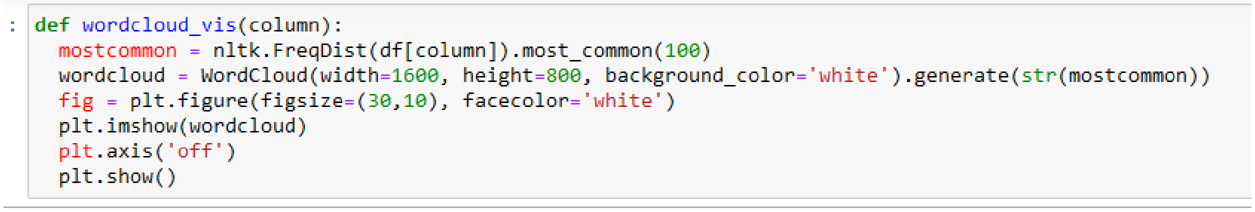
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f73a64e7850>
```

Add Layers (LSTM, Dense-(Hidden Layers), Output)

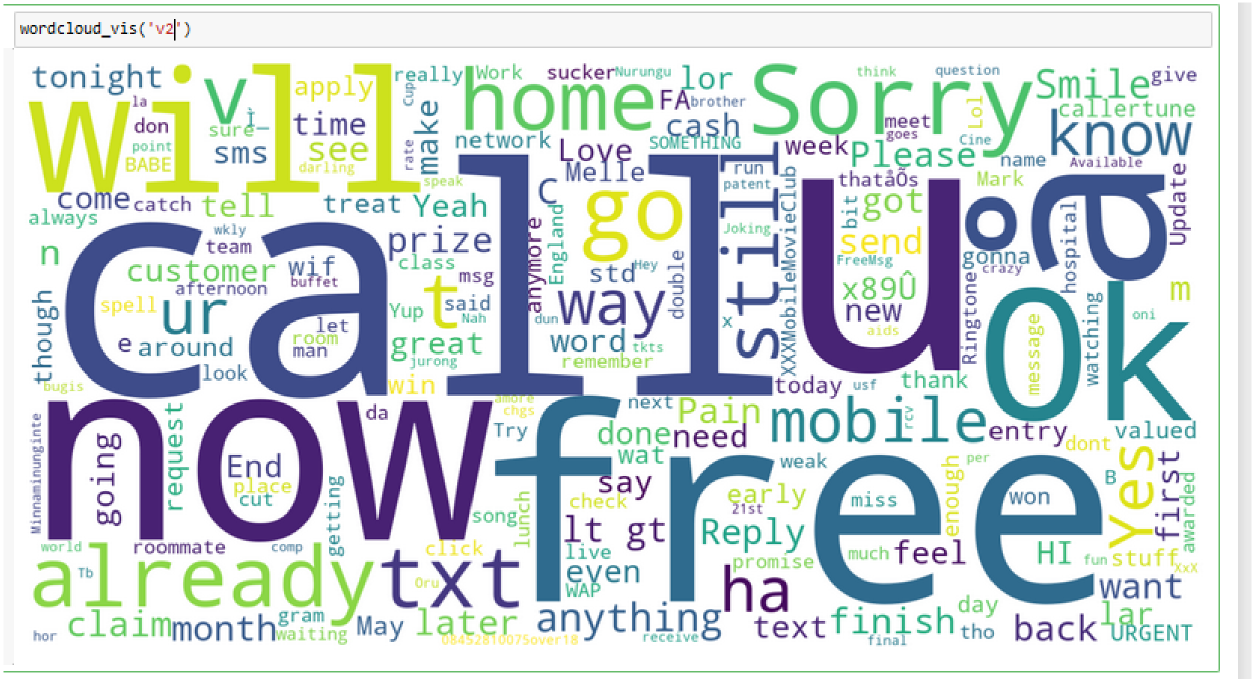
Solution:

```
def wordcloud_vis(column):
  most common = nltk.FreqDist(df[column]).most_common(100)
  word cloud = WordCloud(width=1600, height=800,
background_color='white').generate(str(most common))
  fig = plt.figure(figsize=(30,10), facecolor='white')
  plt.imshow(wordcloud)
```

plt.axis('off')
plt.show()
wordcloud_vis('v2')

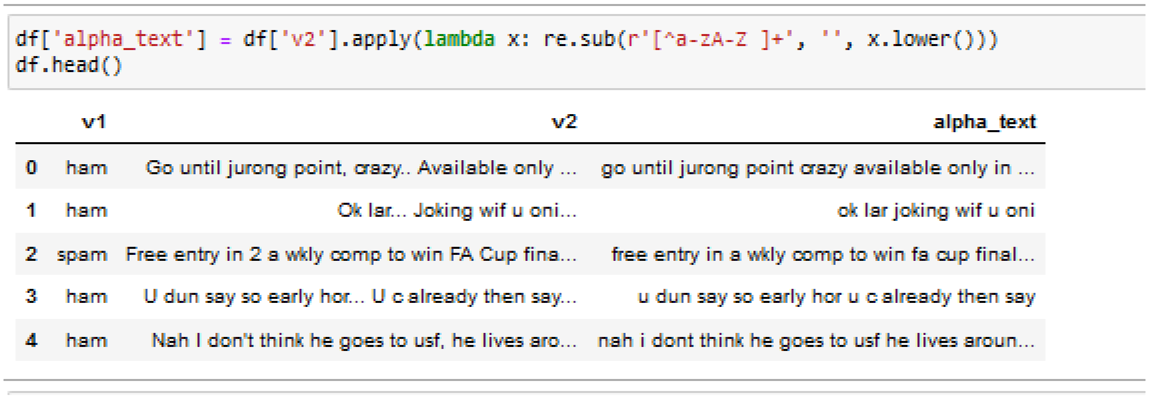


```
: def wordcloud_vis(column):
      mostcommon = nltk.FreqDist(df[column]).most_common(100)
      wordcloud = WordCloud(width=1600, height=800, background_color='white').generate(str(mostcommon))
      fig = plt.figure(figsize=(30,10), facecolor='white')
      plt.imshow(wordcloud)
      plt.axis('off')
      plt.show()
```



```
wordcloud_vis('v2')
```

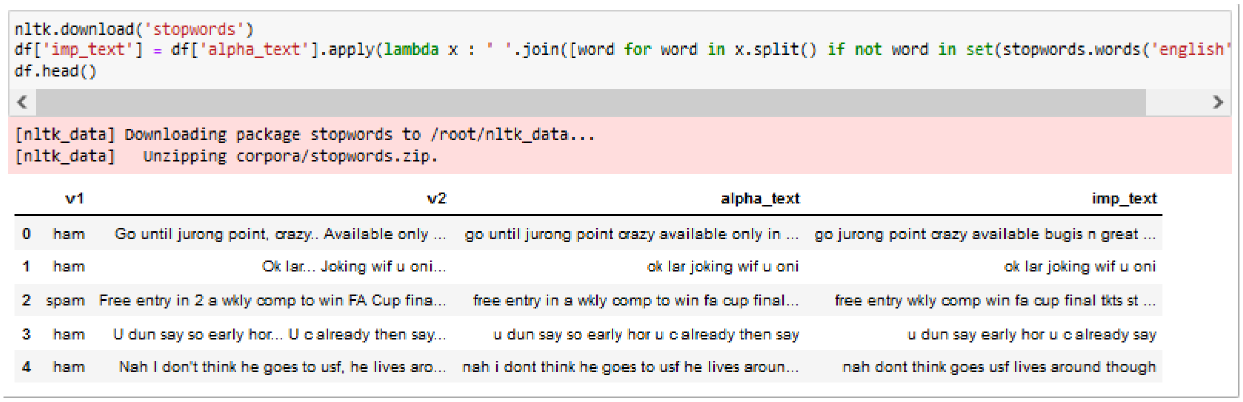
df['alpha_text'] = df['v2'].apply(lambda x: re.sub(r'[^a-zA-Z]+', '', x.lower()))
df.head()



```
df['alpha_text'] = df['v2'].apply(lambda x: re.sub(r'[^a-zA-Z ]+', '', x.lower()))
df.head()
```

	v1	v2	alpha_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...
3	ham	U dun say so early hor... U c already then say...	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...	nah i dont think he goes to usf he lives aroun...

```
nltk.download('stopwords')
df['imp_text'] = df['alpha_text'].apply(lambda x : ' '.join([word for word in x.split() if not word in set(stopwords.words('english'))]))
df.head()
```

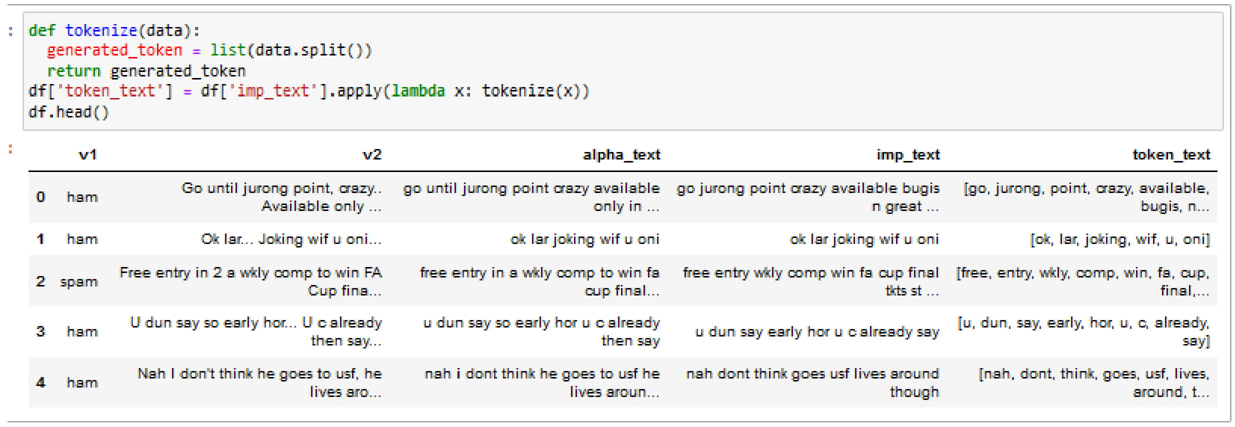


```
nltk.download('stopwords')
df['imp_text'] = df['alpha_text'].apply(lambda x : ' '.join([word for word in x.split() if not word in set(stopwords.words('english'
df.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

	v1	v2	alpha_text	imp_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though

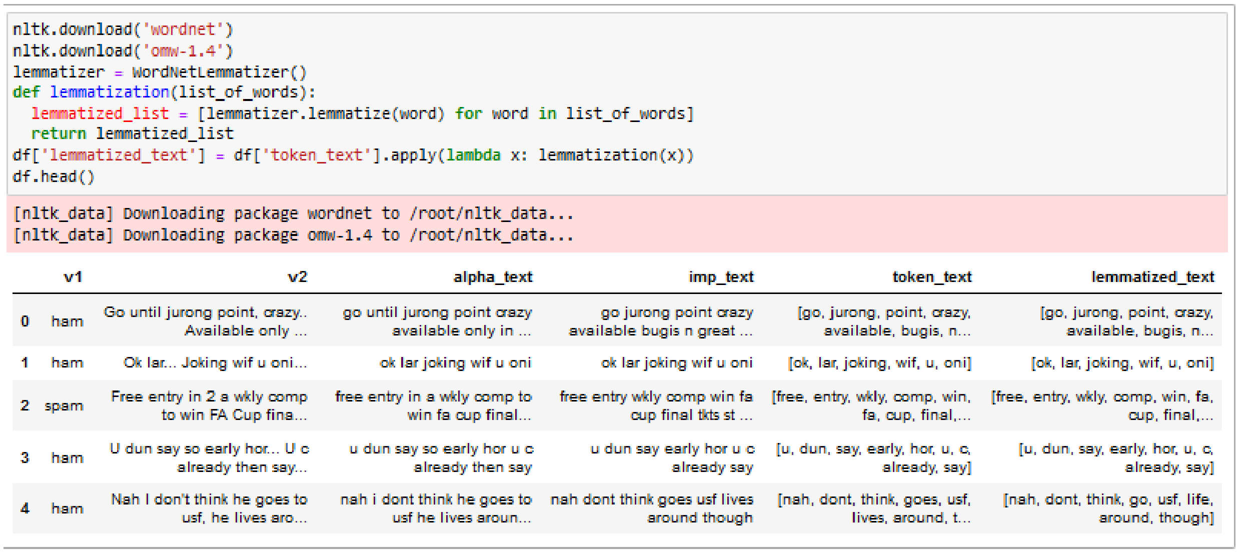
```
def tokenize(data):
  generated_token = list(data.split())
  return generated_token
df['token_text'] = df['imp_text'].apply(lambda x: tokenize(x))
df.head()
```



```
def tokenize(data):
  generated_token = list(data.split())
  return generated_token
df['token_text'] = df['imp_text'].apply(lambda x: tokenize(x))
df.head()
```

	v1	v2	alpha_text	imp_text	token_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...	[go, jurong, point, crazy, available, bugis, n...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say	[u, dun, say, early, hor, u, c, already, say]
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though	[nah, dont, think, goes, usf, lives, around, t...

```
nltk.download('wordnet')
nltk.download('omw-1.4')
lemmatizer = WordNetLemmatizer()
def lemmatization(list_of_words):
  lemmatized_list = [lemmatizer.lemmatize(word) for word in list_of_words]
  return lemmatized_list
df['lemmatized_text'] = df['token_text'].apply(lambda x: lemmatization(x))
df.head()
```

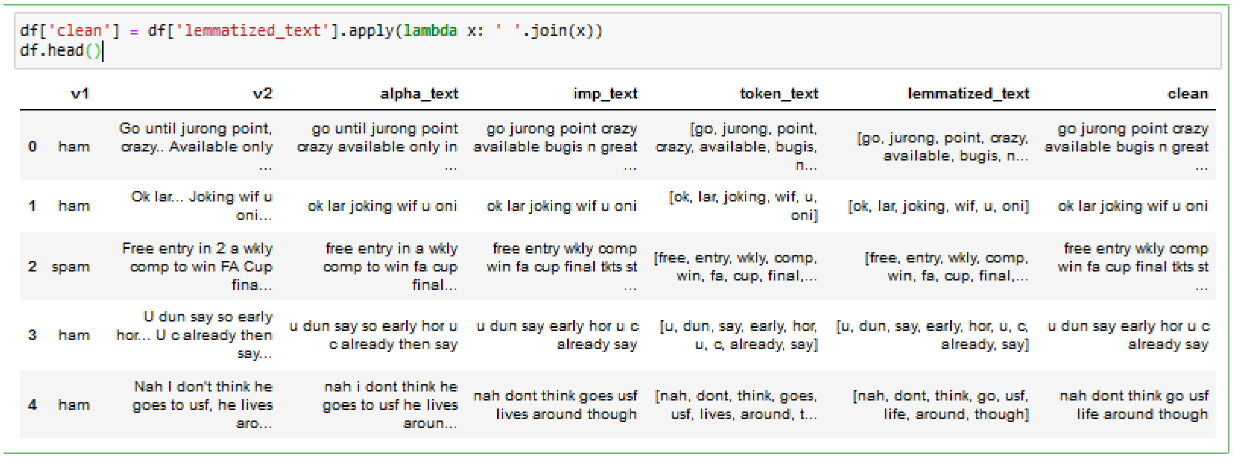


```
nltk.download('wordnet')
nltk.download('omw-1.4')
lemmatizer = WordNetLemmatizer()
def lemmatization(list_of_words):
  lemmatized_list = [lemmatizer.lemmatize(word) for word in list_of_words]
  return lemmatized_list
df['lemmatized_text'] = df['token_text'].apply(lambda x: lemmatization(x))
df.head()
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

	v1	v2	alpha_text	imp_text	token_text	lemmatized_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...	[go, jurong, point, crazy, available, bugis, n...	[go, jurong, point, crazy, available, bugis, n...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]	[ok, lar, joking, wif, u, oni]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...	[free, entry, wkly, comp, win, fa, cup, final,...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say	[u, dun, say, early, hor, u, c, already, say]	[u, dun, say, early, hor, u, c, already, say]
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though	[nah, dont, think, goes, usf, lives, around, t...	[nah, dont, think, go, usf, life, around, though]

df['clean'] = df['lemmatized_text'].apply(lambda x: ' '.join(x))
df.head()



```
df['clean'] = df['lemmatized_text'].apply(lambda x: ' '.join(x))
df.head()
```

	v1	v2	alpha_text	imp_text	token_text	lemmatized_text	clean
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...	[go, jurong, point, crazy, available, bugis, n...	[go, jurong, point, crazy, available, bugis, n...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]	[ok, lar, joking, wif, u, oni]	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...	[free, entry, wkly, comp, win, fa, cup, final,...	[free, entry, wkly, comp, win, fa, cup, final,...	free entry wkly comp win fa cup final tkts st ...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say	[u, dun, say, early, hor, u, c, already, say]	[u, dun, say, early, hor, u, c, already, say]	u dun say early hor u c already say
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though	[nah, dont, think, goes, usf, lives, around, t...	[nah, dont, think, go, usf, life, around, though]	nah dont think go usf life around though

pre-processing

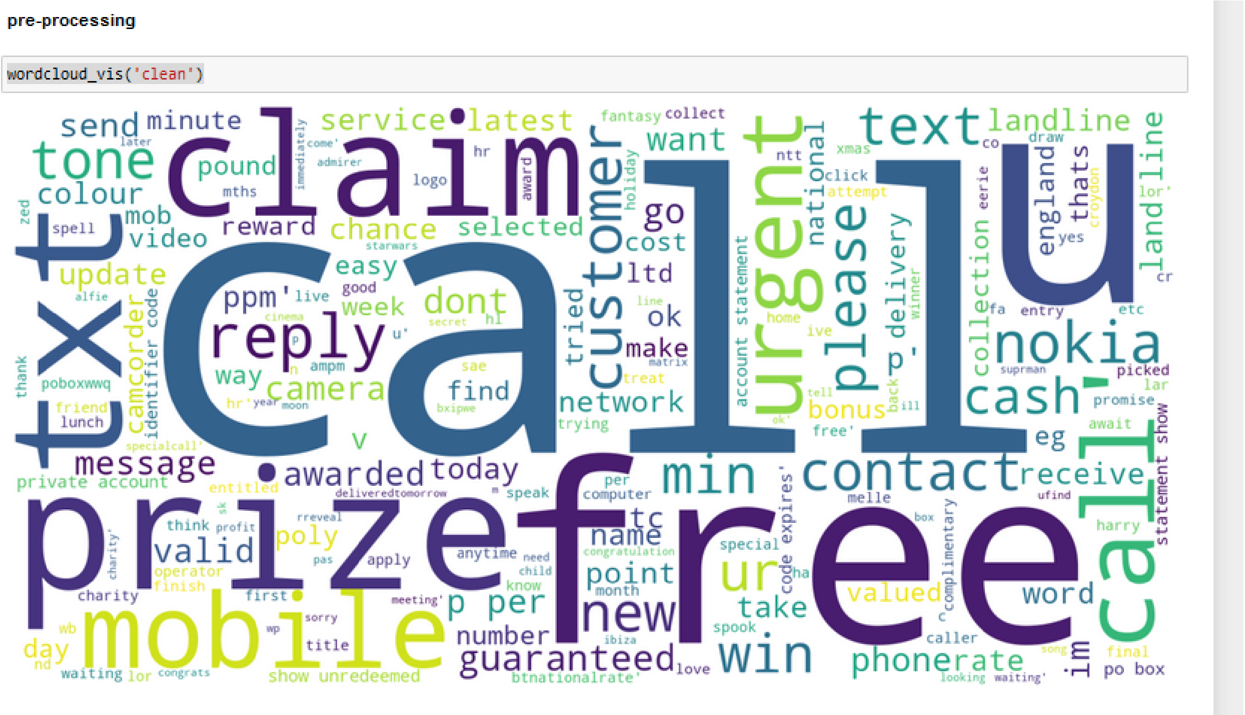
Solution:

wordcloud_vis('clean')
df1 = df.loc[df['v1'] == 'spam']
df2 = df.loc[df['v1'] == 'ham']

spam = set()
df1['clean'].str.lower().str.split().apply(spam.update)
print("Number of unique words in spam", len(spam))

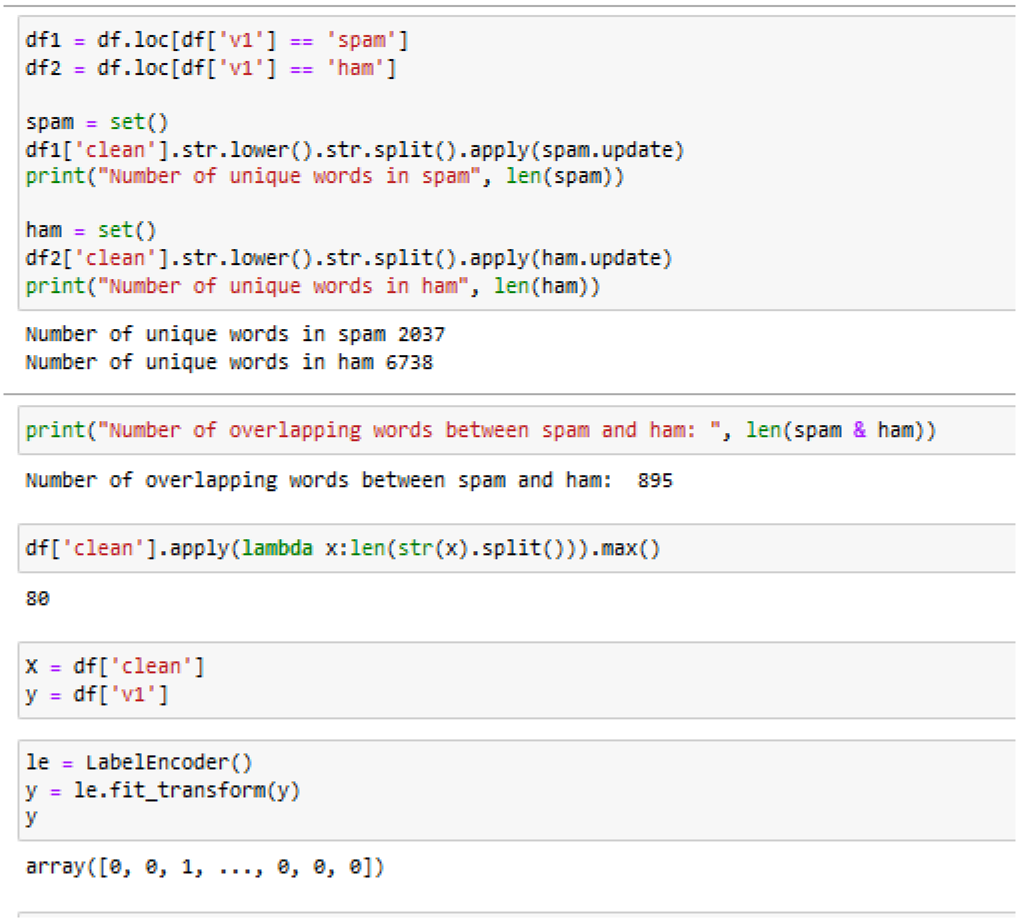
ham = set()

```
df2['clean'].str.lower().str.split().apply(ham.update)
print("Number of unique words in ham", len(ham))
print("Number of overlapping words between spam and ham: ", len(spam & ham))
df['clean'].apply(lambda x:len(str(x).split())).max()
X = df['clean']
y = df['v1']
le = LabelEncoder()
y = le.fit_transform(y)
Y
```



pre-processing

```
wordcloud_vis('clean')
```



```
df1 = df.loc[df['v1'] == 'spam']
df2 = df.loc[df['v1'] == 'ham']

spam = set()
df1['clean'].str.lower().str.split().apply(spam.update)
print("Number of unique words in spam", len(spam))

ham = set()
df2['clean'].str.lower().str.split().apply(ham.update)
print("Number of unique words in ham", len(ham))
```

```
Number of unique words in spam 2037
Number of unique words in ham 6738
```

```
print("Number of overlapping words between spam and ham: ", len(spam & ham))
```

```
Number of overlapping words between spam and ham:  895
```

```
df['clean'].apply(lambda x:len(str(x).split())).max()
```

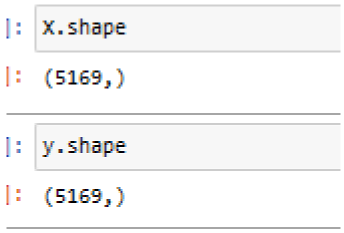
```
80
```

```
X = df['clean']
y = df['v1']
```

```
le = LabelEncoder()
y = le.fit_transform(y)
y
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

X.shape
y.shape



```
]: X.shape
]: (5169,)
```

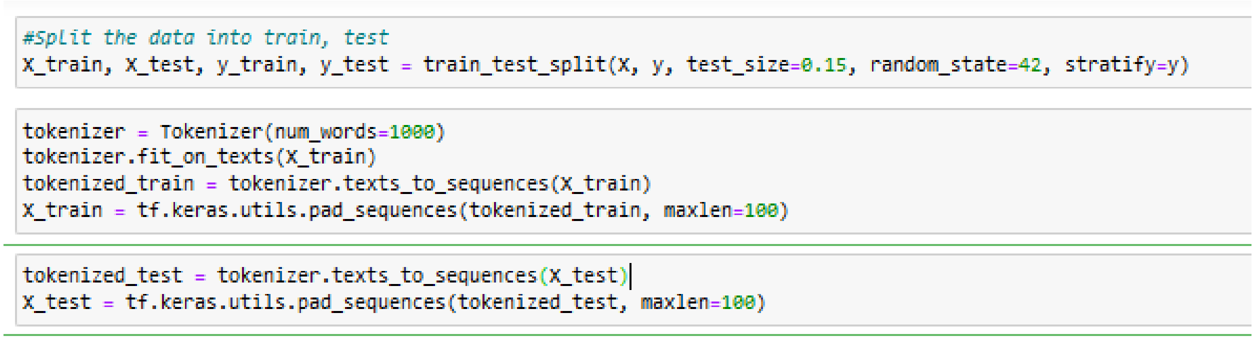
```
]: y.shape
]: (5169,)
```

TEST THE MODEL

Solution:

#Split the data into train, test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=42, stratify=y)
tokenizer = Tokenizer(num_words=1000)
tokenizer.fit_on_texts(X_train)
tokenized_train = tokenizer.texts_to_sequences(X_train)

X_train = tf.keras.utils.pad_sequences(tokenized_train, maxlen=100)
tokenized_test = tokenizer.texts_to_sequences(X_test)
X_test = tf.keras.utils.pad_sequences(tokenized_test, maxlen=100)



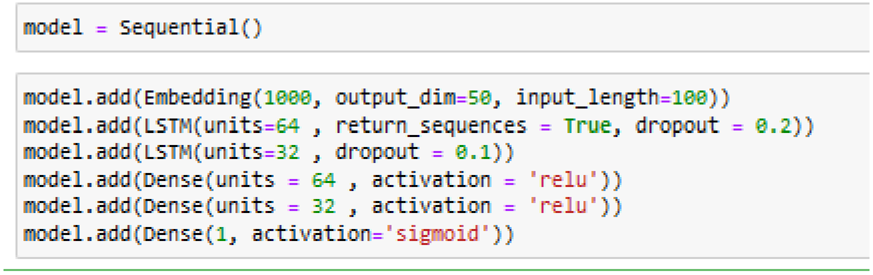
```
#Split the data into train, test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=42, stratify=y)
```

```
tokenizer = Tokenizer(num_words=1000)
tokenizer.fit_on_texts(X_train)
tokenized_train = tokenizer.texts_to_sequences(X_train)
X_train = tf.keras.utils.pad_sequences(tokenized_train, maxlen=100)
```

```
tokenized_test = tokenizer.texts_to_sequences(X_test)
X_test = tf.keras.utils.pad_sequences(tokenized_test, maxlen=100)
```

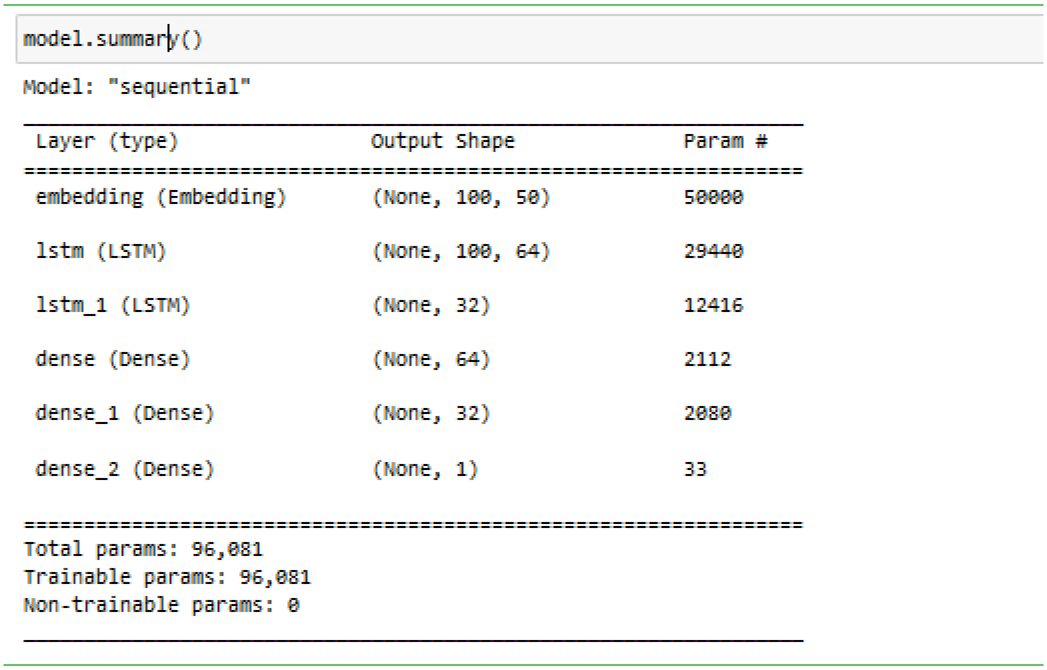
CREATE THE MODEL

Solution:
model = Sequential()
model.add(Embedding(1000, output_dim=50, input_length=100))
model.add(LSTM(units=64 , return_sequences = True, dropout = 0.2))
model.add(LSTM(units=32 , dropout = 0.1))
model.add(Dense(units = 64 , activation = 'relu'))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()



```
model = Sequential()
```

```
model.add(Embedding(1000, output_dim=50, input_length=100))
model.add(LSTM(units=64 , return_sequences = True, dropout = 0.2))
model.add(LSTM(units=32 , dropout = 0.1))
model.add(Dense(units = 64 , activation = 'relu'))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
```

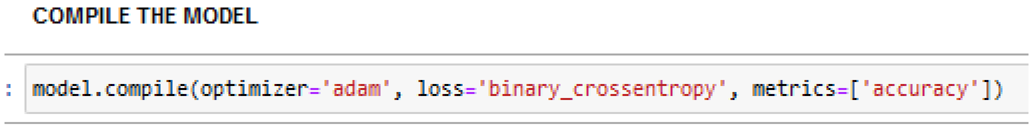


```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 50)           50000

 lstm (LSTM)                 (None, 100, 64)           29440

 lstm_1 (LSTM)               (None, 32)                12416

 dense (Dense)               (None, 64)                2112

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 96,081
Trainable params: 96,081
Non-trainable params: 0
_________________________________________________________________
```

COMPILE THE MODEL

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

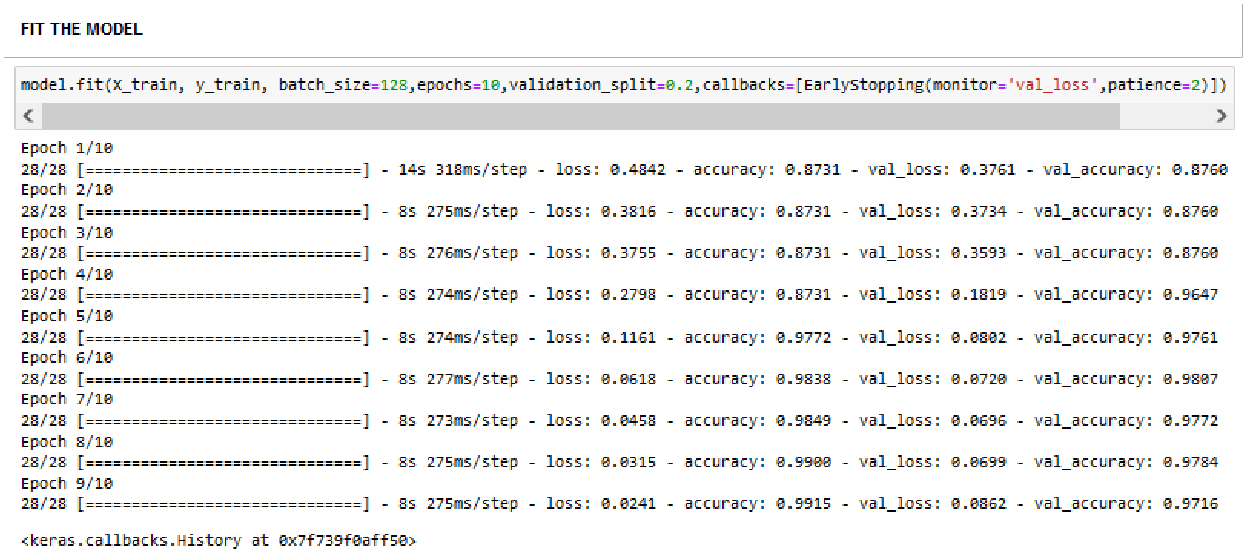


COMPILE THE MODEL

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

FIT THE MODEL

Solution:
model.fit(X_train, y_train, batch_size=128,epochs=10,validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',patience=2)])



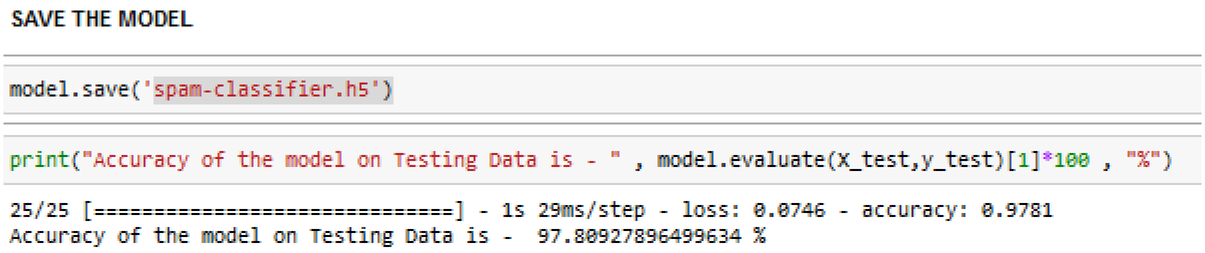
FIT THE MODEL

```
model.fit(X_train, y_train, batch_size=128,epochs=10,validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',patience=2)])
```

```
Epoch 1/10
28/28 [==============================] - 14s 318ms/step - loss: 0.4842 - accuracy: 0.8731 - val_loss: 0.3761 - val_accuracy: 0.8760
Epoch 2/10
28/28 [==============================] - 8s 275ms/step - loss: 0.3816 - accuracy: 0.8731 - val_loss: 0.3734 - val_accuracy: 0.8760
Epoch 3/10
28/28 [==============================] - 8s 276ms/step - loss: 0.3755 - accuracy: 0.8731 - val_loss: 0.3593 - val_accuracy: 0.8760
Epoch 4/10
28/28 [==============================] - 8s 274ms/step - loss: 0.2798 - accuracy: 0.8731 - val_loss: 0.1819 - val_accuracy: 0.9647
Epoch 5/10
28/28 [==============================] - 8s 274ms/step - loss: 0.1161 - accuracy: 0.9772 - val_loss: 0.0802 - val_accuracy: 0.9761
Epoch 6/10
28/28 [==============================] - 8s 277ms/step - loss: 0.0618 - accuracy: 0.9838 - val_loss: 0.0720 - val_accuracy: 0.9807
Epoch 7/10
28/28 [==============================] - 8s 273ms/step - loss: 0.0458 - accuracy: 0.9849 - val_loss: 0.0696 - val_accuracy: 0.9772
Epoch 8/10
28/28 [==============================] - 8s 275ms/step - loss: 0.0315 - accuracy: 0.9900 - val_loss: 0.0699 - val_accuracy: 0.9784
Epoch 9/10
28/28 [==============================] - 8s 275ms/step - loss: 0.0241 - accuracy: 0.9915 - val_loss: 0.0862 - val_accuracy: 0.9716

<keras.callbacks.History at 0x7f739f0aff50>
```

SAVE THE MODEL

Solution:
model.save('spam-classifier.h5')
print("Accuracy of the model on Testing Data is - " , model.evaluate(X_test,y_test)[1]*100 , "%")



SAVE THE MODEL

```
model.save('spam-classifier.h5')
```

```
print("Accuracy of the model on Testing Data is - " , model.evaluate(X_test,y_test)[1]*100 , "%")
```

```
25/25 [==============================] - 1s 29ms/step - loss: 0.0746 - accuracy: 0.9781
Accuracy of the model on Testing Data is -  97.80927896499634 %
```