```
# import libraries
try:
  # %tensorflow_version only exists in Colab.
  !pip install tf-nightly
except Exception:
  pass
import tensorflow as tf
import pandas as pd
from tensorflow import keras
!pip install tensorflow-datasets
import tensorflow_datasets as tfds
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)


# get data files
!wget https://cdn.freecodecamp.org/project-data/sms/train-data.tsv
!wget https://cdn.freecodecamp.org/project-data/sms/valid-data.tsv

train_file_path = "train-data.tsv"
test_file_path = "valid-data.tsv"
```

```
--2024-12-20 10:00:57--  https://cdn.freecodecamp.org/project-data/sms/train-data.tsv
Resolving cdn.freecodecamp.org (cdn.freecodecamp.org)... 104.26.3.33, 104.26.2.33, 172.67.70.149, ...
Connecting to cdn.freecodecamp.org (cdn.freecodecamp.org)|104.26.3.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 358233 (350K) [text/tab-separated-values]
Saving to: ‘train-data.tsv.2’

train-data.tsv.2    100%[===================>] 349.84K  --.-KB/s    in 0.03s

2024-12-20 10:00:57 (9.95 MB/s) - ‘train-data.tsv.2’ saved [358233/358233]

--2024-12-20 10:00:57--  https://cdn.freecodecamp.org/project-data/sms/valid-data.tsv
Resolving cdn.freecodecamp.org (cdn.freecodecamp.org)... 104.26.3.33, 104.26.2.33, 172.67.70.149, ...
Connecting to cdn.freecodecamp.org (cdn.freecodecamp.org)|104.26.3.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 118774 (116K) [text/tab-separated-values]
Saving to: ‘valid-data.tsv.2’

valid-data.tsv.2    100%[===================>] 115.99K  --.-KB/s    in 0.02s

2024-12-20 10:00:57 (5.92 MB/s) - ‘valid-data.tsv.2’ saved [118774/118774]
```

```
import string

import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression


nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
df = pd.read_csv("train-data.tsv", sep="\t")
df.columns = ['label', 'text']
df
```

	label	text
0	ham	you can never do nothing
1	ham	now u sound like manky scouse boy steve,like! ...
2	ham	mum say we wan to go then go... then she can s...
3	ham	never y lei... i v lazy... got wat? dat day ü ...
4	ham	in xam hall boy asked girl tell me the startin...
...	...	...
4173	ham	just woke up. yeesh its late. but i didn't fal...
4174	ham	what do u reckon as need 2 arrange transport i...
4175	spam	free entry into our £250 weekly competition ju...
4176	spam	-pls stop bootydelious (32/f) is inviting you ...
4177	ham	tell my bad character which u dnt lik in me. ...

4178 rows × 2 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df['text'] = df['text'].apply(lambda x: x.replace('\r\n', ' '))
df
df.text.iloc[1]
```

```
stemmer = PorterStemmer() # reduce words to their root form (e.g., "running" to "run").
corpus = [] # an empty list to store the processed text data

# Creates a set of common English stopwords (e.g., "the", "and", "is") that will be removed from the text.
stopwords_set = set(stopwords.words('english'))

for i in range(len(df)):
    text = df['text'].iloc[i].lower() # lower case all the text

    # remove all punctuations and splits it into individual words
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = text.split()

    # Applies stemming to each word in the text and removes stopwords.
    text = [stemmer.stem(word) for word in text if word not in stopwords_set]

    text = ' '.join(text)
    corpus.append(text)


# # Initialize the CountVectorizer
# vectorizer = CountVectorizer()

# # Transform the corpus into a document-term matrix
# X = vectorizer.fit_transform(corpus).toarray()
# Y = df.label #FIXED

# FIXED
vectorizer = TfidfVectorizer(ngram_range=(1, 2))  # Using bi-grams
X = vectorizer.fit_transform(corpus)
Y = df['label'].apply(lambda x: 1 if x == 'spam' else 0)

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)


# clf = RandomForestClassifier(n_jobs=-1)
clf = LogisticRegression(solver='liblinear')
clf.fit(X_train, Y_train)
```

```
LogisticRegression(solver='liblinear')
```

```
clf.score(X_test, Y_test)
```

```
0.9102870813397129
```

```
email_to_classify = df.text.values[10]
email_text = email_to_classify.lower().translate(str.maketrans('','',string.punctuation)).split()
email_text = [stemmer.stem(word) for word in text if word not in stopwords_set] # Stem and Remove Stopwords
email_text = ' '.join(email_text) # Combines the processed words back into a single string.

email_corpus = [email_text] # Puts the processed email text into a list.

X_email = vectorizer.transform(email_corpus)


clf.predict(X_email)

df.label.iloc[10]

# the prediction is correct
```

```
'ham'
```

```
def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = text.split()
    text = [stemmer.stem(word) for word in text if word not in stopwords_set]
    return ' '.join(text)
```

You should create a function called predict_message that takes a message string as an argument and returns a list. The first element in the list should be a number between zero and one that indicates the likeliness of "ham" (0) or "spam" (1). The second element in the list should be the word "ham" or "spam", depending on which is most likely.

For this challenge, you will use the SMS Spam Collection dataset. The dataset has already been grouped into train data and test data.

The first two cells import the libraries and data. The final cell tests your model and function. Add your code in between these cells.

```
# function to predict messages based on model
# (should return list containing prediction and label, ex. [0.008318834938108921, 'ham'])
def predict_message(pred_text):
    pred_text = pred_text.lower()
    pred_text = pred_text.translate(str.maketrans('', '', string.punctuation))
    pred_text = pred_text.split()
    pred_text = [stemmer.stem(word) for word in pred_text if word not in stopwords_set]
    pred_text = ' '.join(pred_text)

    X_pred = vectorizer.transform([pred_text])
    prediction_probability = clf.predict_proba(X_pred)[0][1]
    prediction_label = "spam" if prediction_probability > 0.5 else "ham"
    result = [prediction_probability, prediction_label]
    print(result)
    return result

# pred_text = "how are you doing today?"

# prediction = predict_message(pred_text)
# print(prediction)

test_messages = ["how are you doing today",
                 "sale today! to stop texts call 98912460324",
                 "i dont want to go. can we try it a different day? available sat",
                 "our new mobile video service is live. just install on your phone to start watching.",
                 "you have won £1000 cash! call to claim your prize.",
                 "i'll bring it tomorrow. don't forget the milk.",
                 "wow, is your arm alright. that happened to me one time too"
                ]

for msg in test_messages:
    print(msg)
    prediction = predict_message(msg)
```

```
how are you doing today
[0.0961879512054695, 'ham']
sale today! to stop texts call 98912460324
[0.7037496096806443, 'spam']
```

```
      i dont want to go. can we try it a different day? available sat
      [0.049362429650175475, 'ham']
      our new mobile video service is live. just install on your phone to start watching.
      [0.5081622620083248, 'spam']
      you have won £1000 cash! call to claim your prize.
      [0.7692762522114522, 'spam']
      i'll bring it tomorrow. don't forget the milk.
      [0.0587232399080203, 'ham']
      wow, is your arm alright. that happened to me one time too
      [0.05522518134953335, 'ham']


# Run this cell to test your function and model. Do not modify contents.
def test_predictions():
  test_messages = ["how are you doing today",
                   "sale today! to stop texts call 98912460324",
                   "i dont want to go. can we try it a different day? available sat",
                   "our new mobile video service is live. just install on your phone to start watching.",
                   "you have won £1000 cash! call to claim your prize.",
                   "i'll bring it tomorrow. don't forget the milk.",
                   "wow, is your arm alright. that happened to me one time too"
                  ]

  test_answers = ["ham", "spam", "ham", "spam", "spam", "ham", "ham"]
  passed = True

  for msg, ans in zip(test_messages, test_answers):
    prediction = predict_message(msg)
    if prediction[1] != ans:
      passed = False

  if passed:
    print("You passed the challenge. Great job!")
  else:
    print("You haven't passed yet. Keep trying.")

test_predictions()
```

```
[0.0961879512054695, 'ham']
[0.7037496096806443, 'spam']
[0.049362429650175475, 'ham']
[0.5081622620083248, 'spam']
[0.7692762522114522, 'spam']
[0.0587232399080203, 'ham']
[0.05522518134953335, 'ham']
You passed the challenge. Great job!
```