Step 0. Unzip enron1.zip into the current directory.

Step 1. Traverse the dataset and create a Pandas dataframe. This is already done for you and should run without any errors. You should recognize Pandas from task 1.

In [4]:
```
import pandas as pd
import os

def read_spam():
    category = 'spam'
    directory = './enron1/spam'
    return read_category(category, directory)

def read_ham():
    category = 'ham'
    directory = './enron1/ham'
    return read_category(category, directory)

def read_category(category, directory):
    emails = []
    for filename in os.listdir(directory):
        if not filename.endswith(".txt"):
            continue
        with open(os.path.join(directory, filename), 'r') as fp:
            try:
                content = fp.read()
                emails.append({'name': filename, 'content': content, 'category': categor
            except:
                print(f'skipped {filename}')
    return emails

ham = read_ham()
spam = read_spam()

df = pd.DataFrame.from_records(ham)
df = df.append(pd.DataFrame.from_records(spam))
```

```
skipped 2649.2004-10-27.GP.spam.txt
skipped 0754.2004-04-01.GP.spam.txt
skipped 2042.2004-08-30.GP.spam.txt
skipped 3304.2004-12-26.GP.spam.txt
skipped 4142.2005-03-31.GP.spam.txt
skipped 3364.2005-01-01.GP.spam.txt
skipped 4201.2005-04-05.GP.spam.txt
skipped 2140.2004-09-13.GP.spam.txt
skipped 2248.2004-09-23.GP.spam.txt
skipped 4350.2005-04-23.GP.spam.txt
skipped 4566.2005-05-24.GP.spam.txt
skipped 2526.2004-10-17.GP.spam.txt
skipped 1414.2004-06-24.GP.spam.txt
skipped 2698.2004-10-31.GP.spam.txt
skipped 5105.2005-08-31.GP.spam.txt
/var/folders/qg/07_8byfx39x_mpyfv_k2p81h0000gn/T/ipykernel_72959/217166261.py:31: Future
Warning: The frame.append method is deprecated and will be removed from pandas in a futu
re version. Use pandas.concat instead.
  df = df.append(pd.DataFrame.from_records(spam))
```

Step 2. Data cleaning is a critical part of machine learning. You and I can recognize that 'Hello' and 'hello' are the same word but a machine does not know this a priori. Therefore, we can 'help' the machine by conducting such normalization steps for it. Write a function `preprocessor` that takes in a string and replaces all non alphabet characters with a space and then lowercases the result.

In [6]:
```
import re

def preprocessor(e):
    return re.sub('[^A-Za-z]', ' ', e).lower()
```

Step 3. We will now train the machine learning model. All the functions that you will need are imported for you. The instructions explain how the work and hint at which functions to use. You will likely need to refer to the scikit learn documentation to see how exactly to invoke the functions. It will be handy to keep that tab open.

In [9]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# The CountVectorizer converts a text sample into a vector (think of it as an array of f
# Each entry in the vector corresponds to a single word and the value is the number of t
# Instantiate a CountVectorizer. Make sure to include the preprocessor you previously wr
vectorizer = CountVectorizer(preprocessor=preprocessor)


# Use train_test_split to split the dataset into a train dataset and a test dataset.
# The machine learning model learns from the train dataset.
# Then the trained model is tested on the test dataset to see if it actually learned any
# If it just memorized for example, then it would have a low accuracy on the test datase
X_train,X_test,y_train,y_test = train_test_split(df["content"],df["category"],test_size=


# Use the vectorizer to transform the existing dataset into a form in which the model ca
# Remember that simple machine learning models operate on numbers, which the CountVector
X_train_df = vectorizer.fit_transform(X_train)


# Use the LogisticRegression model to fit to the train dataset.
# You may remember y = mx + b and Linear Regression from high school. Here, we fitted a
# Logistic Regression is another form of regression.
# However, Logistic Regression helps us determine if a point should be in category A or
model = LogisticRegression()
model.fit(X_train_df,y_train)


# Validate that the model has learned something.
# Recall the model operates on vectors. First transform the test set using the vectorize
# Then generate the predictions.
X_test_df = vectorizer.transform(X_test)
y_pred = model.predict(X_test_df)


# We now want to see how we have done. We will be using three functions.
# `accuracy_score` tells us how well we have done.
# 90% means that every 9 of 10 entries from the test dataset were predicted accurately.
# The `confusion_matrix` is a 2x2 matrix that gives us more insight.
# The top left shows us how many ham emails were predicted to be ham (that's good!).
# The bottom right shows us how many spam emails were predicted to be spam (that's good!
# The other two quadrants tell us the misclassifications.
# Finally, the `classification_report` gives us detailed statistics which you may have s
print(f'Accuracy:\n{accuracy_score(y_test,y_pred)}\n')
print(f'Confusion Matrix:\n{confusion_matrix(y_test,y_pred)}\n')
print(f'Detailed Statistics:\n{classification_report(y_test,y_pred)}\n')
```

```
Accuracy:
0.9767441860465116
```

```
Confusion Matrix:
[[726  16]
 [  8 282]]

Detailed Statistics:
              precision    recall  f1-score   support

         ham       0.99      0.98      0.98       742
        spam       0.95      0.97      0.96       290

    accuracy                           0.98      1032
   macro avg       0.97      0.98      0.97      1032
weighted avg       0.98      0.98      0.98      1032
```

```
/Users/philipokonkwo/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Step 4.

In [10]:

```python
# Let's see which features (aka columns) the vectorizer created.
# They should be all the words that were contained in the training dataset.
features = vectorizer.get_feature_names_out()


# You may be wondering what a machine learning model is tangibly. It is just a collectio
# You can access these numbers known as "coefficients" from the coef_ property of the mo
# We will be looking at coef_[0] which represents the importance of each feature.
# What does importance mean in this context?
# Some words are more important than others for the model.
# It's nothing personal, just that spam emails tend to contain some words more frequentl
# This indicates to the model that having that word would make a new email more likely t
importance = model.coef_[0]


# Iterate over importance and find the top 10 positive features with the largest magnitu
# Similarly, find the top 10 negative features with the largest magnitude.
# Positive features correspond to spam. Negative features correspond to ham.
# You will see that `http` is the strongest feature that corresponds to spam emails.
# It makes sense. Spam emails often want you to click on a link.
l = list(enumerate(importance))
print()
l.sort(key=lambda e: e[1], reverse=True)
for i,imp in l[:10]:
    print(imp, features[i])
print()
l.sort(key=lambda e: -e[1], reverse=True)
for i,imp in l[:10]:
    print(imp, features[i])
```

```
1.0612889752388919 http
0.8171515585622918 prices
0.7939426861839024 no
0.7566968582150021 remove
0.7217196205323224 removed
0.6916606935181643 off
0.6795654331849164 hello
0.6537527670765367 mobile
```

```
0.6338503913327512 more
0.6164954004427172 message

-1.54371369555463 attached
-1.5401927124896693 daren
-1.5015981121415158 enron
-1.3008717964901506 thanks
-1.249615045420233 doc
-1.180471504508258 meter
-1.0830238812452668 xls
-1.0658043957514949 hpl
-1.0460339462676282 neon
-1.036751027720975 deal
```

Submission

1. Upload the jupyter notebook to Forage.

All Done!