```
import glob, cv2

name_list = glob.glob('fresh.png')

X = []
name_label = []

for name in name_list:
  name_label.append(0)

  img = cv2.imread(name)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  img = cv2.resize(img, (50, 50))

  X.append((img))

len(X)
```

```
1
```

```
import glob, cv2

name_list = glob.glob('rotten.png')

X = []
name_label = []

for name in name_list:
  name_label.append(0)

  img = cv2.imread(name)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  img = cv2.resize(img, (50, 50))
```

```
  X.append((img))

len(X)
```

```
1
```

```
import numpy as np

X = np.array(X)
y = np.array(name_label)

print(X.shape)
print(y.shape)
```

```
(1, 50, 50)
(1,)
```

```
#Cau 2
```

```
import pandas as pd

data = pd.read_csv('smsspamcollection.tsv', sep='\t')
data.head()
```

	label	message	length	punct
0	ham	Go until jurong point, crazy.. Available only ...	111	9
1	ham	Ok lar... Joking wif u oni...	29	6
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	6
3	ham	U dun say so early hor... U c already then say...	49	6
4	ham	Nah I don't think he goes to usf, he lives aro...	61	2

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

```
data.isnull().sum()
```

	0
label	0
message	0
length	0
punct	0

dtype: int64

```
print(data['label'].value_counts())
data.shape
```

```
label
ham     4825
spam     747
Name: count, dtype: int64
(5572, 4)
```

```
df_ham = data[data.label == 'ham']
test_ham = df_ham.sample(frac=0.2, random_state=0)
print(test_ham.shape)

train_ham = df_ham.drop(test_ham.index)
print(train_ham.shape)

df_spam = data[data.label == 'spam']
test_spam = df_spam.sample(frac=0.2, random_state=0)
```

```
print(test_spam.shape)
train_spam = df_spam.drop(test_spam.index)
print(train_spam.shape)
```

```
(965, 4)
(3860, 4)
(149, 4)
(598, 4)
```

```
train_df = pd.concat([train_ham, train_spam])
print(train_df.shape)

test_df = pd.concat([test_ham, test_spam])
print(test_df.shape)
```

```
(4458, 4)
(1114, 4)
```

```
vocab = []

for msg in train_df.message.values:
  for t in msg.lower().split():
    if t not in vocab:
      vocab.append(t)
```

```
vocab_size = len(vocab)
word2ix = {t : i for i, t in enumerate(vocab)}
X_train = np.zeros((train_df.shape[0], vocab_size))

for i, msg in enumerate(train_df.message.values):
  for t in msg.lower().split():
    if t in vocab:
      X_train[i, word2ix[t]] += 1
```

```
X_train.shape
```

```
(4458, 11970)
```

```
X_test = np.zeros((test_df.shape[0], vocab_size))

for i, msg in enumerate(test_df.message.values):
  for t in msg.lower().split():
    if t in vocab:
      X_test[i, word2ix[t]] += 1

X_test.shape
```

```
(1114, 11970)
```

```
X_min = X_train.min(axis=0, keepdims=True)
X_max = X_train.max(axis=0, keepdims=True)

X_train_scale = (X_train - X_min) / (X_max - X_min)
X_test_scale = (X_test - X_min) / (X_max - X_min)
```

```
y_train = train_df.loc[:, ['label']].values
y_test = test_df.loc[:, ['label']].values
print(y_train.shape, y_test.shape)
```

```
(4458, 1) (1114, 1)
```

```
y_train_new = np.where(y_train == 'spam', 0, 1)
y_test_new = np.where(y_test == 'spam', 0, 1)
y_train_new[:10]
```

```
array([[1],
       [1],
       [1],
```

```
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1]])
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, recall_score, f1_score

model = LogisticRegression()
model.fit(X_train_scale, y_train_new)

y_pred = model.predict(X_test_scale)

acc = accuracy_score(y_test_new, y_pred)
recall = recall_score(y_test_new, y_pred)
f1 = f1_score(y_test_new, y_pred)

print("Accuracy:", acc)
print("Recall:", recall)
print("F1-score:", f1)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vec
  y = column_or_1d(y, warn=True)
Accuracy: 0.9757630161579892
Recall: 0.9989637305699481
F1-score: 0.9861892583120204
```

```
def batch_generator(X, y, batch_size=32, shuffle=True):
    n_samples = X.shape[0]
    indices = np.arange(n_samples)

    if shuffle:
        np.random.shuffle(indices)
```

```
        np.random.shuffle(indices)

    for start in range(0, n_samples, batch_size):
        end = start + batch_size
        batch_idx = indices[start:end]
        yield X[batch_idx], y[batch_idx]
```

```
class binary1:
    def __init__(self, lr=0.01, epochs=100, batch_size=32):
        self.lr = lr
        self.epochs = epochs
        self.batch_size = batch_size
        self.weights = None
        self.bias = None

    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    def fit(self, X, y):
        n_samples, n_features = X.shape

        self.weights = np.zeros(n_features)
        self.bias = 0

        for epoch in range(self.epochs):
            for X_batch, y_batch in batch_generator(X, y, self.batch_size):

                linear = np.dot(X_batch, self.weights) + self.bias
                y_pred = self.sigmoid(linear)

                dw = (1 / len(y_batch)) * np.dot(X_batch.T, (y_pred - y_batch))
                db = (1 / len(y_batch)) * np.sum(y_pred - y_batch)

                self.weights -= self.lr * dw
                self.bias -= self.lr * db

    def predict(self, X):
```

```
    def predict(self, X):
        linear = np.dot(X, self.weights) + self.bias
        y_pred = self.sigmoid(linear)
        return np.where(y_pred >= 0.5, 1, 0)

    def evaluate(self, X, y):
        y_pred = self.predict(X)
        acc = accuracy_score(y, y_pred)
        recall = recall_score(y, y_pred)
        f1 = f1_score(y, y_pred)
        return acc, recall, f1

if __name__ == "__main__":
    model = binary1(lr=0.1, epochs=100, batch_size=16)
    model.fit(X_train_scale, y_train_new)

    acc, recall, f1 = model.evaluate(X_test_scale, y_test_new)
    print(f"Accuracy: {acc:.4f}")
    print(f"Recall:   {recall:.4f}")
    print(f"F1-score: {f1:.4f}")
```

```
Accuracy: 0.9600
Recall:   0.9545
F1-score: 0.9545
```