```
import gzip
import numpy as np

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, recall_score, f1_score
```

```
#1. 读取 MNIST 数据：加载并处理手写数字图像数据及其对应的标签。
#读取图像数据函数
def load_images(filename):
    with gzip.open(filename, 'rb') as f:
        magic_number = int.from_bytes(f.read(4), 'big')
        num_images = int.from_bytes(f.read(4), 'big')
        num_rows = int.from_bytes(f.read(4), 'big')
        num_cols = int.from_bytes(f.read(4), 'big')
        images = np.frombuffer(f.read(), dtype=np.uint8)
        images = images.reshape(num_images, num_rows * num_cols)
        return images
```

```
# 读取标签数据函数
def load_labels(filename):
    with gzip.open(filename, 'rb') as f:
        magic_number = int.from_bytes(f.read(4), 'big')
        num_labels = int.from_bytes(f.read(4), 'big')
        labels = np.frombuffer(f.read(), dtype=np.uint8)
        return labels
```

```
#2. 划分训练集和测试集：将图像数据划分为训练集和测试集，比例一般为7:3或8（由于下载好的数据集已经划分了训练集和测试集，我们这里只需要分别读取）
# 加载训练集数据和标签
X_train = load_images('/Users/zhangpuchang/Downloads/DM3/data/train-images-idx3-ubyte.gz')
y_train = load_labels('/Users/zhangpuchang/Downloads/DM3/data/train-labels-idx1-ubyte.gz')

# 加载测试集数据和标签
X_test = load_images('/Users/zhangpuchang/Downloads/DM3/data/t10k-images-idx3-ubyte.gz')
y_test = load_labels('/Users/zhangpuchang/Downloads/DM3/data/t10k-labels-idx1-ubyte.gz')
```

```
#3. 特征缩放：对图像数据进行归一化处理，缩放到[0,1]范围。
def norm_images(images):
    return images/255.0
```

```
X_train = norm_images(X_train)
X_test = norm_images(X_test)
print('X_train shape:', X_train.shape)
print('y_train shape:', y_train.shape)
print('X_test shape:', X_test.shape)
print('y_test shape:', y_test.shape)
```

```
X_train shape: (60000, 784)
y_train shape: (60000,)
X_test shape: (10000, 784)
y_test shape: (10000,)
```

```
#4. 构建支持向量机模型：选择适当的核函数（如线性核、多项式核、径向基核等），训练模型。
#5. 模型评估：在测试集上进行预测，计算模型的准确率、召回率、F 1 值等指标，评估模型性能。
#我们分别评估不同核函数的分类性能

#线性核
svm_linear = SVC(kernel='linear', C=1.0, random_state=42)
svm_linear.fit(X_train, y_train)
```

```
y_pred = svm_linear.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print("线性核")
print('准确率：', accuracy)
print('召回率：', recall)
print('f1：', f1)
```

```
线性核
准确率： 0.9404
召回率： 0.9394223313176161
f1： 0.9395137440765534
```

```
#多项式核
svm_poly = SVC(kernel='poly', C=1.0, random_state=42)
svm_poly.fit(X_train, y_train)
```

```
y_pred = svm_poly.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print("多项式核")
print('准确率：', accuracy)
print('召回率：', recall)
print('f1：', f1)
```

```
多项式核
准确率： 0.9771
召回率： 0.9769380249358539
f1： 0.9770040605295229
```

```
#高斯核
svm_rbf = SVC(kernel='rbf', C=1.0, random_state=42)
svm_rbf.fit(X_train, y_train)
```

```
y_pred = svm_rbf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print("高斯核")
print('准确率：', accuracy)
print('召回率：', recall)
print('f1：', f1)
```

```
高斯核
准确率： 0.9792
召回率： 0.9790919842945065
f1： 0.9791298259748042
```

```
#sigmoid核
svm_sigmoid = SVC(kernel='sigmoid', C=1.0, random_state=42)
svm_sigmoid.fit(X_train, y_train)
```

```
y_pred = svm_sigmoid.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print("sigmoid核")
print('准确率：', accuracy)
print('召回率：', recall)
print('f1：', f1)
```

```
sigmoid核
准确率： 0.7759
召回率： 0.772509593691401
f1： 0.7722759380621314
```