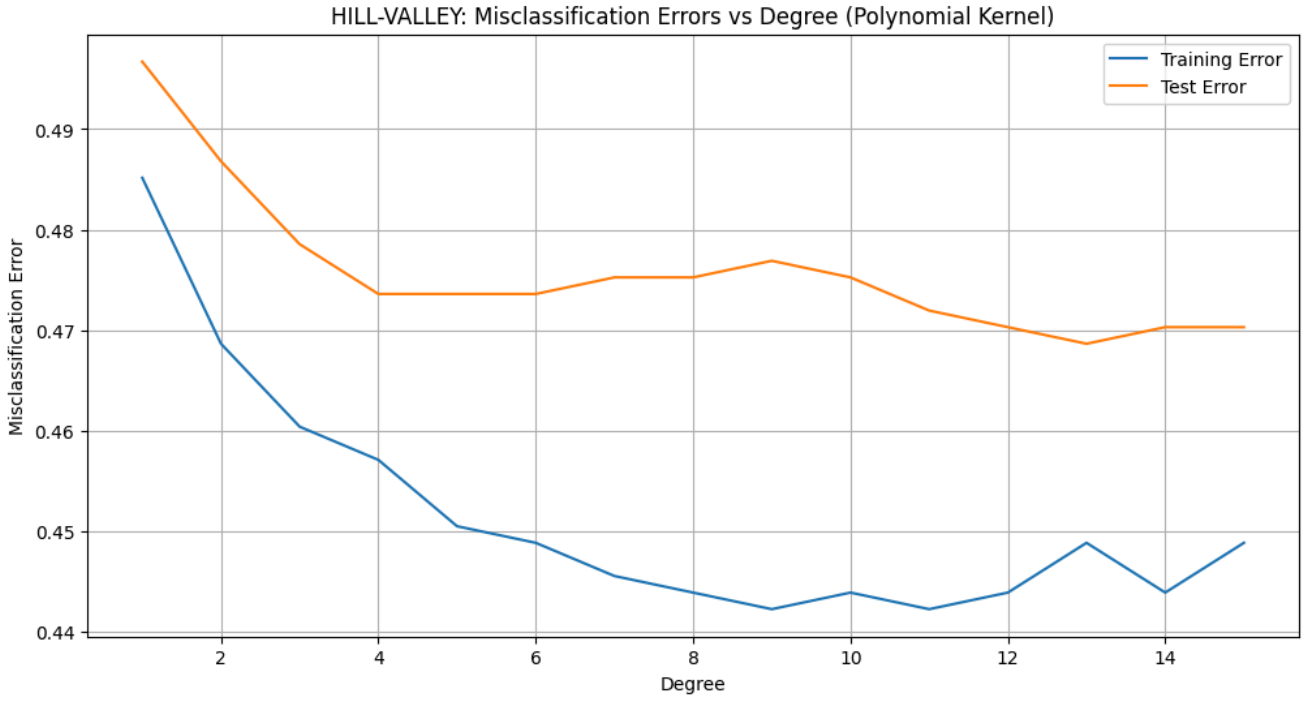
In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

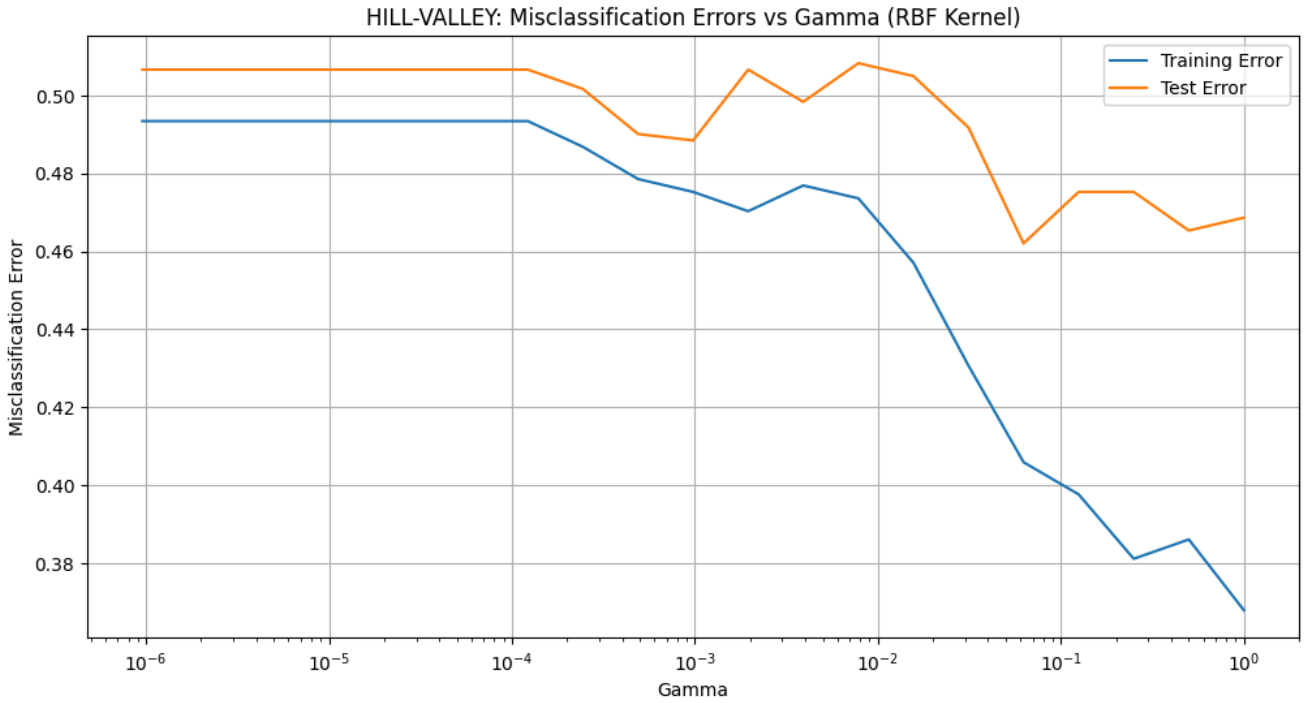
In [ ]:
```
X_train = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Hill-valley/X.dat', sep = ' ',  header=None)
y_train = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Hill-valley/Y.dat')
X_test = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Hill-valley/Xtest.dat', sep = ' ',  header=None)
y_test = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Hill-valley/Ytest.dat')

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [ ]:
```
degrees = range(1, 16)
train_errors = []
test_errors = []

for d in degrees:
    svm = SVC(kernel = 'poly', degree = d)
    svm.fit(X_train_scaled, y_train)
    train_pred = svm.predict(X_train_scaled)
    test_pred = svm.predict(X_test_scaled)
    train_errors.append(1 - accuracy_score(y_train, train_pred))
    test_errors.append(1 - accuracy_score(y_test, test_pred))

plt.figure(figsize = (12, 6))
plt.plot(degrees, train_errors, label = 'Training Error')
plt.plot(degrees, test_errors, label = 'Test Error')
plt.title('HILL-VALLEY: Misclassification Errors vs Degree (Polynomial Kernel)')
plt.xlabel('Degree')
plt.ylabel('Misclassification Error')
plt.legend()
plt.grid(True)
plt.show()
```



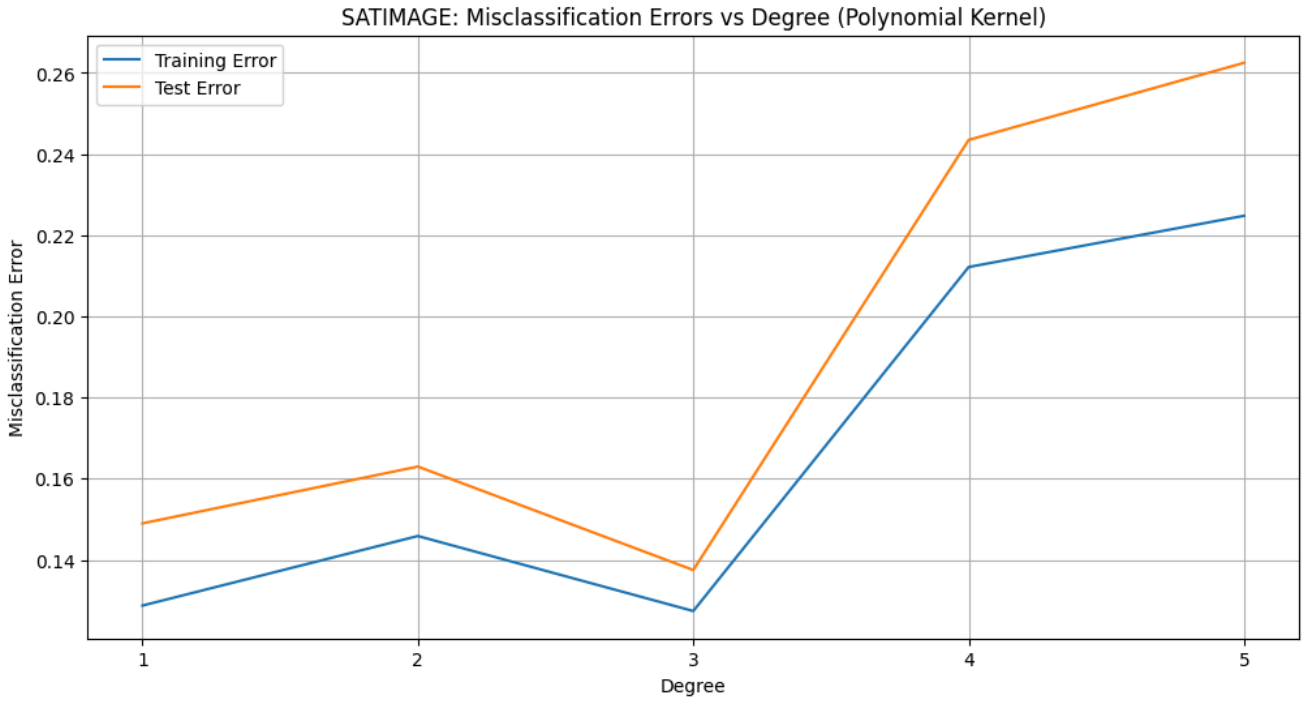
In [ ]:
```
gammas = np.logspace(0, -20, num=21, base=2)
train_errors = []
test_errors = []

for g in gammas:
    svm = SVC(gamma = g)
    svm.fit(X_train_scaled, y_train)
    train_pred = svm.predict(X_train_scaled)
    test_pred = svm.predict(X_test_scaled)
    train_errors.append(1 - accuracy_score(y_train, train_pred))
    test_errors.append(1 - accuracy_score(y_test, test_pred))

plt.figure(figsize = (12, 6))
plt.semilogx(gammas, train_errors, label = 'Training Error')
plt.semilogx(gammas, test_errors, label = 'Test Error')
```

```
plt.title('HILL-VALLEY: Misclassification Errors vs Gamma (RBF Kernel)')
plt.xlabel('Gamma')
plt.ylabel('Misclassification Error')
plt.legend()
plt.grid(True)
plt.show()
```



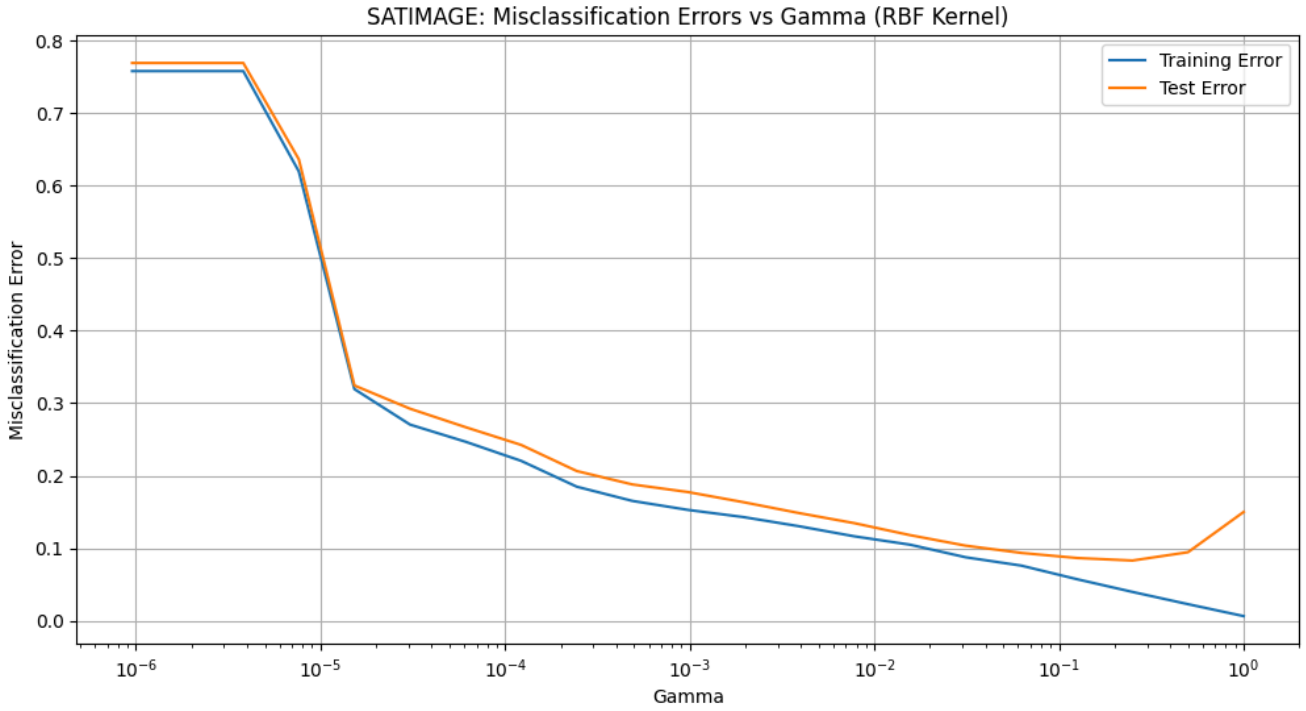
In [ ]:
```
X_train = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/satimage/X.dat', sep = ' ',  header=None)
y_train = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/satimage/Y.dat')
X_test = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/satimage/Xtest.dat', sep = ' ',  header=None)
y_test = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/satimage/Ytest.dat')

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [ ]:
```
degrees = range(1, 6)
train_errors = []
test_errors = []

for d in degrees:
    svm = SVC(kernel = 'poly', degree = d)
    svm.fit(X_train_scaled, y_train)
    train_pred = svm.predict(X_train_scaled)
    test_pred = svm.predict(X_test_scaled)
    train_errors.append(1 - accuracy_score(y_train, train_pred))
    test_errors.append(1 - accuracy_score(y_test, test_pred))

plt.figure(figsize = (12, 6))
plt.plot(degrees, train_errors, label = 'Training Error')
plt.plot(degrees, test_errors, label = 'Test Error')
plt.title('SATIMAGE: Misclassification Errors vs Degree (Polynomial Kernel)')
plt.xlabel('Degree')
plt.ylabel('Misclassification Error')
plt.legend()
plt.xticks(range(1,6))
plt.grid(True)
plt.show()
```



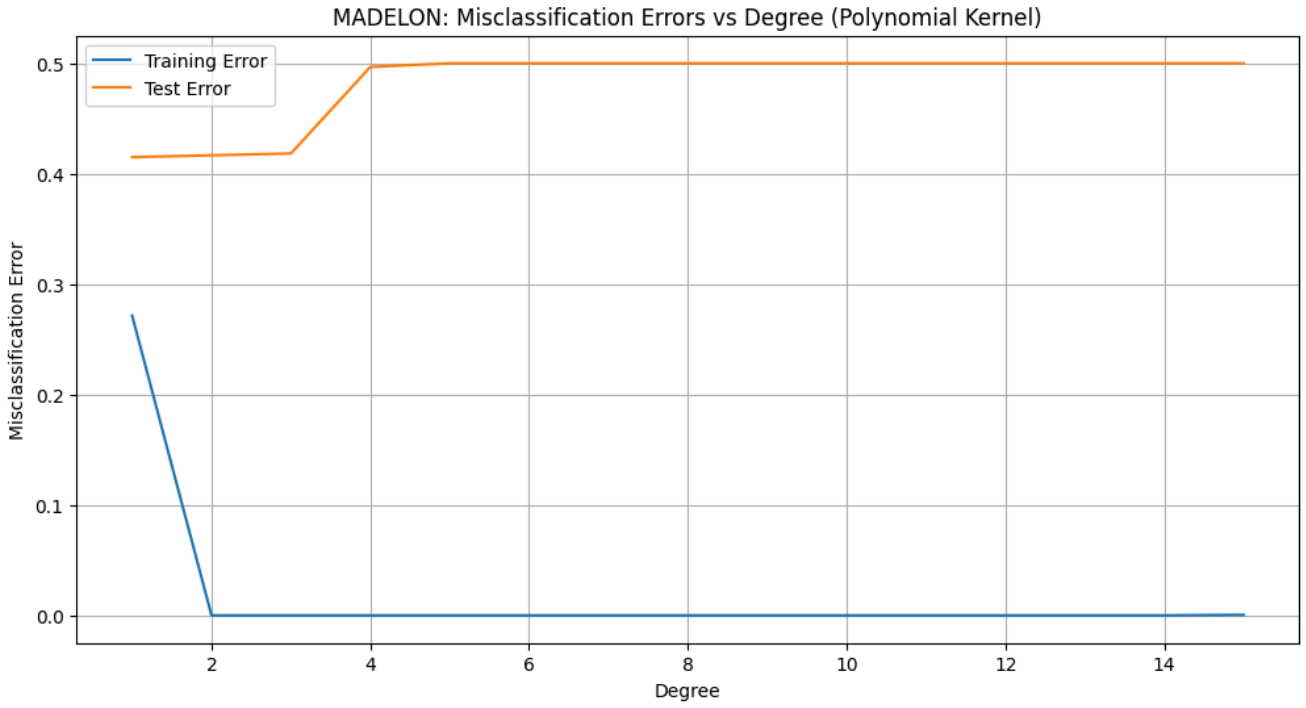
In [ ]:

```
gammas = np.logspace(0, -20, num = 21, base = 2)
train_errors = []
test_errors = []

for g in gammas:
    svm = SVC(gamma = g)
    svm.fit(X_train_scaled, y_train)
    train_pred = svm.predict(X_train_scaled)
    test_pred = svm.predict(X_test_scaled)
    train_errors.append(1 - accuracy_score(y_train, train_pred))
    test_errors.append(1 - accuracy_score(y_test, test_pred))

plt.figure(figsize = (12, 6))
plt.semilogx(gammas, train_errors, label = 'Training Error')
plt.semilogx(gammas, test_errors, label = 'Test Error')
plt.title('SATIMAGE: Misclassification Errors vs Gamma (RBF Kernel)')
plt.xlabel('Gamma')
plt.ylabel('Misclassification Error')
plt.legend()
plt.grid(True)
plt.show()
```



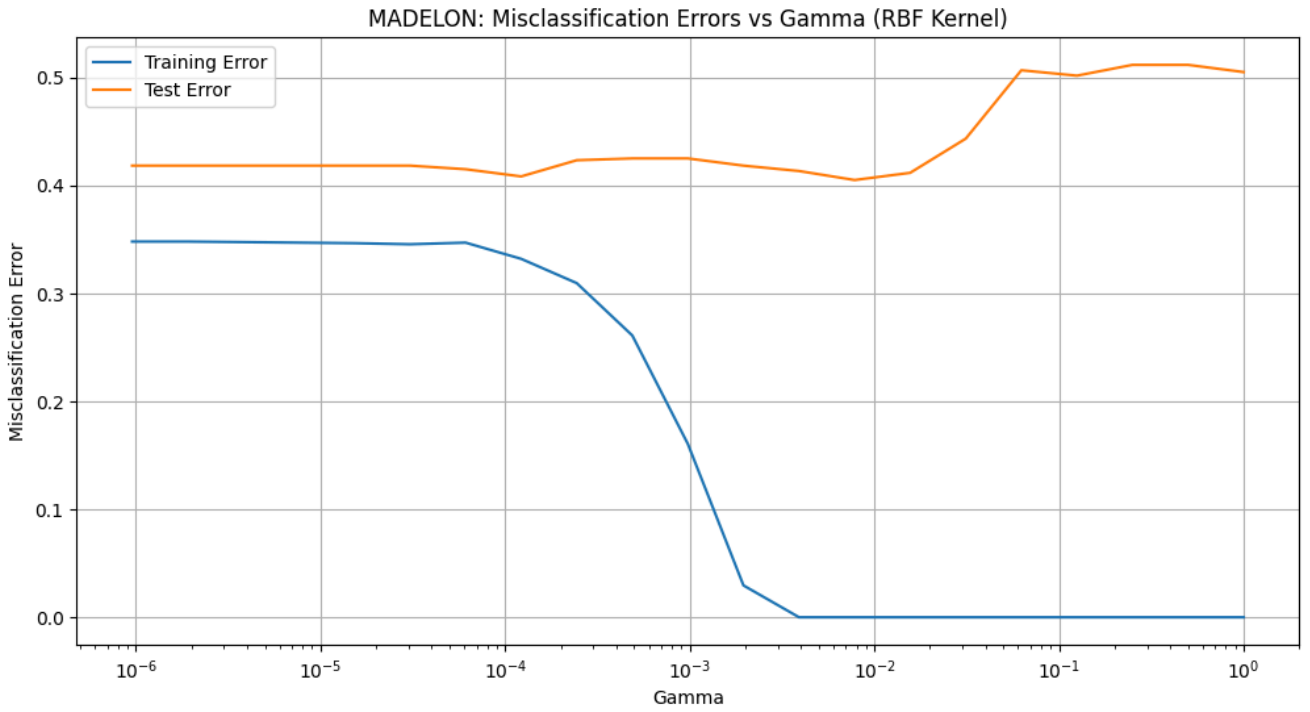
In [ ]:
```
X_train = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/MADELON/madelon_train.data', sep = ' ',  header=None)
y_train = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/MADELON/madelon_train.labels')
X_test = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/MADELON/madelon_valid.data', sep = ' ',  header=None)
y_test = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/MADELON/madelon_valid.labels')

X_train = X_train.drop(500, axis = 1)
X_test = X_test.drop(500, axis = 1)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [ ]:
```
degrees = range(1, 16)
train_errors = []
test_errors = []

for d in degrees:
    svm = SVC(kernel = 'poly',degree = d)
    svm.fit(X_train_scaled, y_train)
    train_pred = svm.predict(X_train_scaled)
    test_pred = svm.predict(X_test_scaled)
    train_errors.append(1 - accuracy_score(y_train, train_pred))
    test_errors.append(1 - accuracy_score(y_test, test_pred))

plt.figure(figsize = (12, 6))
plt.plot(degrees, train_errors, label = 'Training Error')
plt.plot(degrees, test_errors, label = 'Test Error')
plt.title('MADELON: Misclassification Errors vs Degree (Polynomial Kernel)')
plt.xlabel('Degree')
plt.ylabel('Misclassification Error')
plt.legend()
plt.grid(True)
plt.show()
```



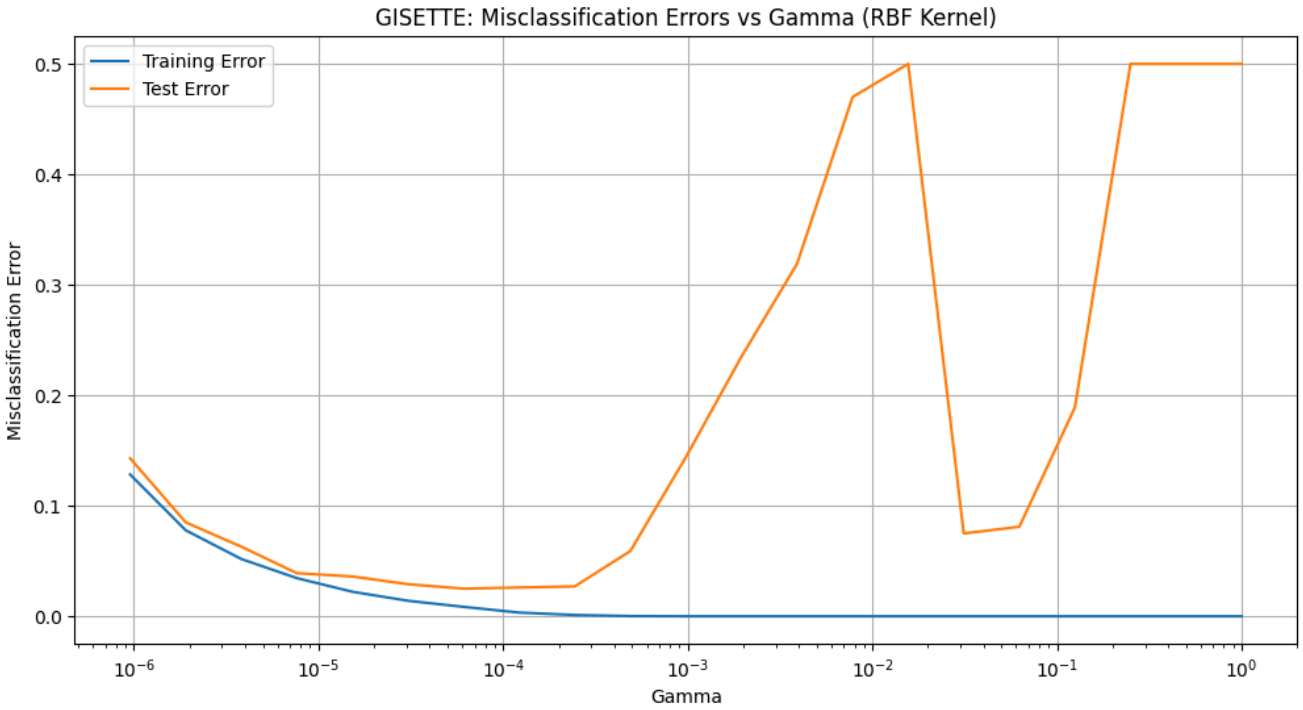
In [ ]:
```
gammas = np.logspace(0, -20, num = 21, base = 2)
train_errors = []
test_errors = []

for g in gammas:
    svm = SVC(gamma = g)
    svm.fit(X_train_scaled, y_train)
    train_pred = svm.predict(X_train_scaled)
    test_pred = svm.predict(X_test_scaled)
    train_errors.append(1 - accuracy_score(y_train, train_pred))
    test_errors.append(1 - accuracy_score(y_test, test_pred))

plt.figure(figsize = (12, 6))
plt.semilogx(gammas, train_errors, label = 'Training Error')
plt.semilogx(gammas, test_errors, label = 'Test Error')
plt.title('MADELON: Misclassification Errors vs Gamma (RBF Kernel)')
plt.xlabel('Gamma')
plt.ylabel('Misclassification Error')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



In [ ]:
```
X_train = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Gisette/gisette_train.data', sep = ' ',  header=None)
y_train = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Gisette/gisette_train.labels')
X_test = pd.read_csv('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Gisette/gisette_valid.data', sep = ' ',  header=None)
y_test = np.loadtxt('D:/School/Applied ML FSU/Applied-ML-FSU/Data/Gisette/gisette_valid.labels')

X_train = X_train.drop(5000, axis = 1)
X_test = X_test.drop(5000, axis = 1)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [ ]:
```
gammas = np.logspace(0, -20, num = 21, base = 2)
train_errors = []
test_errors = []

for g in gammas:
    svm = SVC(gamma = g)
    svm.fit(X_train_scaled, y_train)
    train_pred = svm.predict(X_train_scaled)
    test_pred = svm.predict(X_test_scaled)
    train_errors.append(1 - accuracy_score(y_train, train_pred))
    test_errors.append(1 - accuracy_score(y_test, test_pred))

plt.figure(figsize = (12, 6))
plt.semilogx(gammas, train_errors, label = 'Training Error')
plt.semilogx(gammas, test_errors, label = 'Test Error')
plt.title('GISETTE: Misclassification Errors vs Gamma (RBF Kernel)')
plt.xlabel('Gamma')
plt.ylabel('Misclassification Error')
plt.legend()
plt.grid(True)
plt.show()
```


GISETTE: Misclassification Errors vs Gamma (RBF Kernel)
Training Error
Test Error
Misclassification Error
0.0
0.1
0.2
0.3
0.4
0.5
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
Gamma