In [1]:
```
import numpy as np
from svm import*
from svmutil import *
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
data_train = pd.read_csv("data_train.csv")
data_test = pd.read_csv("data_test.csv")
X = data_train[['intensity','symmetry']]
X_test = data_test[['intensity','symmetry']]
```

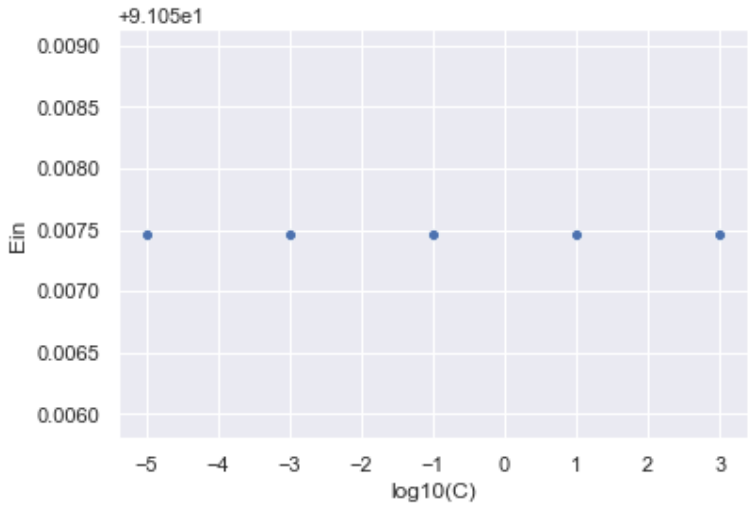
In [3]:
```
y4 = np.where(data_train["digit"] ==4, 1 ,-1)
y4_test = np.where(data_test["digit"] ==4, 1 ,-1)
Clist = [-5,-3,-1,1,3]
```

In [4]:
```
prob = svm_problem(y4,X.values)
param = svm_parameter()
param.kernel_type = POLY
param.coef0 =1
param.degree = 2
model = svm_train(prob, param)
```

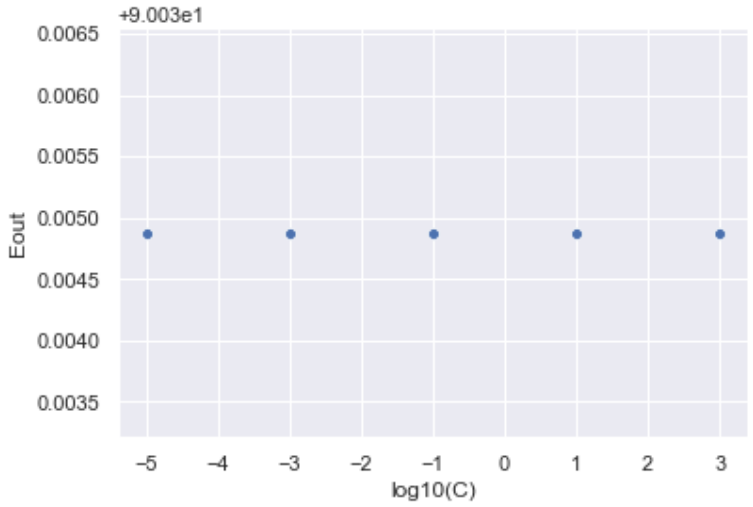
In [5]:
```
Ein = []
Eout = []
for c in Clist:
    param.C = 10**c
    model = svm_train(prob, param)
    Ein = Ein +[svm_predict(y4,X.values,model)[1][0]]
    Eout = Eout +[svm_predict(y4_test,X_test.values,model)[1][0]]
```

```
Accuracy = 91.0575% (6639/7291) (classification)
Accuracy = 90.0349% (1807/2007) (classification)
Accuracy = 91.0575% (6639/7291) (classification)
Accuracy = 90.0349% (1807/2007) (classification)
Accuracy = 91.0575% (6639/7291) (classification)
Accuracy = 90.0349% (1807/2007) (classification)
Accuracy = 91.0575% (6639/7291) (classification)
Accuracy = 90.0349% (1807/2007) (classification)
Accuracy = 91.0575% (6639/7291) (classification)
Accuracy = 90.0349% (1807/2007) (classification)
```

In [6]:
```
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
df=pd.DataFrame({'log10(C)': Clist, 'Ein': Ein,'Eout':Eout})
ax = sns.scatterplot(x='log10(C)', y='Ein',data=df)
```



In [7]:
```
ax = sns.scatterplot(x='log10(C)', y='Eout',data=df)
```



Q14: No matter how C changes, Ein and Eout remains the same. This suggest that Hard-margin SVM does a good job.

In [ ]:
```
```