In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import numpy as np
import pandas as pd

from tqdm import tqdm
from joblib import Parallel, delayed
from os import cpu_count
from svmutil import *
```

In [3]:
```
def problem14():
    trainY, trainX = svm_read_problem("satimage.scale")
    df = pd.DataFrame(trainX).fillna(0)
    trainX, trainY = df.to_numpy(), [1 if i == 3 else -1 for i in trainY]
    model = svm_train(trainY, trainX, "-s 0 -t 0 -c 10")
    index = [i-1 for i in model.get_sv_indices()]
    anyn = np.array(model.get_sv_coef())
    zn = trainX[index]
    print(f"||w|| is {np.linalg.norm(np.dot(anyn.T, zn)):.1f}")


problem14()
```

```
||w|| is 8.5
```

In [4]:
```
def problem15():
    trainY, trainX = svm_read_problem("satimage.scale")
    df = pd.DataFrame(trainX).fillna(0)
    for number in range(1, 6):
        X, Y = df.to_numpy(), [1 if i == number else -1 for i in trainY]
        model = svm_train(Y, X, "-s 0 -t 1 -d 2 -c 10")
        p_label, p_acc, p_val = svm_predict(Y, X, model, "-q")
        ACC, MSE, SCC = evaluations(Y, p_label)
        print(f"Ein of \"{number}\" versus \"not {number}\" is {ACC:.2f}")


problem15()
```

```
Ein of "1" versus "not 1" is 98.38
Ein of "2" versus "not 2" is 99.30
Ein of "3" versus "not 3" is 95.81
Ein of "4" versus "not 4" is 90.64
Ein of "5" versus "not 5" is 95.65
```

In [5]:
```
def problem17():
    trainY, trainX = svm_read_problem("satimage.scale")
    df = pd.DataFrame(trainX).fillna(0)
    for number in range(1, 6):
        X, Y = df.to_numpy(), [1 if i == number else -1 for i in trainY]
        model = svm_train(Y, X, "-s 0 -t 1 -d 2 -c 10")
        sv = model.get_nr_sv()
        print(
            f"The number of support vector(s) for \"{number}\" versus \"not {number}\"
```

In [6]:
```
def problem18():
    trainY, trainX = svm_read_problem("satimage.scale")
    X, Y = trainX, [1 if i == 6 else -1 for i in trainY]
```

```
    testY, testX = svm_read_problem("satimage.scale.t")
    testY = [1 if i == 6 else -1 for i in testY]

    for cost in [0.01, 0.1, 1, 10, 100]:
        model = svm_train(Y, X, f"-s 0 -t 2 -g 10 -c {cost}")
        p_label, p_acc, p_val = svm_predict(testY, testX, model, "-q")
        ACC, MSE, SCC = evaluations(testY, p_label)
        print(f"Eout of C={cost} is {ACC:.2f}")


problem18()
```

```
Eout of C=0.01 is 76.50
Eout of C=0.1 is 83.65
Eout of C=1 is 89.35
Eout of C=10 is 90.30
Eout of C=100 is 90.30
```

In [7]:

```
def problem19():
    trainY, trainX = svm_read_problem("satimage.scale")
    X, Y = trainX, [1 if i == 6 else -1 for i in trainY]

    testY, testX = svm_read_problem("satimage.scale.t")
    testY = [1 if i == 6 else -1 for i in testY]

    for gamma in [0.1, 1, 10, 100, 1000]:
        model = svm_train(Y, X, f"-s 0 -t 2 -g {gamma} -c 0.1")
        p_label, p_acc, p_val = svm_predict(testY, testX, model, "-q")
        ACC, MSE, SCC = evaluations(testY, p_label)
        print(f"Eout of gamma={gamma} is {ACC:.2f}")


problem19()
```

```
Eout of gamma=0.1 is 90.15
Eout of gamma=1 is 93.00
Eout of gamma=10 is 83.65
Eout of gamma=100 is 76.50
Eout of gamma=1000 is 76.50
```

In [9]:

```
def problem20():
    resultDict = {0.1: [], 1: [], 10: [], 100: [], 1000: []}

    trainY, trainX = svm_read_problem("satimage.scale")
    X, Y = pd.DataFrame(trainX).fillna(0), pd.Series(
        [1 if i == 6 else -1 for i in trainY])

    for _ in tqdm(range(1000)):
        indexes = np.random.choice(X.index, size=200)
        learnX, learnY = X.drop(indexes).to_numpy(), Y.drop(indexes).to_numpy()
        valX, valY = X.loc[indexes].to_numpy(), Y.loc[indexes].to_numpy()
        result = Parallel(n_jobs=cpu_count())(delayed(run)(
            learnX, learnY, valX, valY, gamma) for gamma in [0.1, 1, 10, 100, 1000])
        for key, value in result:
            resultDict[key].append(value)

    for gamma, score in resultDict.items():
        print(f"Gamma={gamma}, Eval is {sum(score) / len(score):.2f}")


def run(learnX: np.ndarray, learnY: np.array, valX: np.ndarray, valY: np.array, gamma:
    model = svm_train(learnY, learnX, f"-s 0 -t 2 -g {gamma} -c 0.1")
```

```
    p_label, p_acc, p_val = svm_predict(valY, valX, model, "-q")
    ACC, MSE, SCC = evaluations(valY, p_label)
    return (gamma, ACC)


problem20()
```

```
100%|██████████| 1000/1000 [1:04:32<00:00,  3.87s/it]Gamma=0.1, Eval is 91.84
Gamma=1, Eval is 92.85
Gamma=10, Eval is 83.40
Gamma=100, Eval is 76.46
Gamma=1000, Eval is 76.46
```