```
# Questions 2-4
from sklearn import svm
from sklearn import cross_validation
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np

C = 0.01
Q = 2

target_digit = 2

test = np.loadtxt("features.test.txt")
train = np.loadtxt("features.train.txt")

digit = []

all_others = []

for row in train:
    if row[0] == target_digit:
        digit.append([row[0], row[1], row[2]])
    else:
        all_others.append([row[0], row[1], row[2]])
digit = np.asarray(digit)
all_others = np.asarray(all_others)

trues = np.ones(len(digit))
falses = np.full((len(all_others), 1), -1)
classifications = np.append(trues, falses)

digit = digit[:, [1, 2]]
all_others = all_others[:, [1,2]]
data = np.append(digit, all_others, axis = 0)

clf = svm.SVC(kernel = 'poly', C=C, degree = Q, coef0 = 1.0, gamma = 1)

clf.fit(data, classifications)

y_pred = clf.predict(data)
comparisons = np.equal(y_pred, classifications)
print 1 - float(np.sum(comparisons)) / len(comparisons)
print len(clf.support_vectors_)
```

```
# Questions 5-6
from sklearn import svm
from sklearn import cross_validation
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np

C = 1
Q = 2

target_digit = 1
vs_digit = 5

test = np.loadtxt("features.test.txt")
train = np.loadtxt("features.train.txt")

digit = []

other = []

for row in train:
    if row[0] == target_digit:
        digit.append([row[0], row[1], row[2]])
    elif row[0] == vs_digit:
        other.append([row[0], row[1], row[2]])
digit = np.asarray(digit)
other = np.asarray(other)

trues = np.ones(len(digit))
falses = np.full((len(other), 1), -1)
classifications = np.append(trues, falses)

digit = digit[:, [1, 2]]
other = other[:, [1,2]]
data = np.append(digit, other, axis = 0)

clf = svm.SVC(kernel = 'poly', C=C, degree = Q, coef0 = 1.0, gamma = 1)

clf.fit(data, classifications)

y_pred = clf.predict(data)
comparisons = np.equal(y_pred, classifications)
print 1 - float(np.sum(comparisons)) / len(comparisons)
print len(clf.support_vectors_)
```

```
# Questions 7-8
from sklearn import svm
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np

C = 0.001
Q = 2
runs = 100
target_digit = 1
vs_digit = 5

ECV = 0

test = np.loadtxt("features.test.txt")
train = np.loadtxt("features.train.txt")


for k in range(runs) :
    digit = []

    other = []

    for row in train:
        if row[0] == target_digit:
            digit.append([row[0], row[1], row[2]])
        elif row[0] == vs_digit:
            other.append([row[0], row[1], row[2]])
    digit = np.asarray(digit)
    other = np.asarray(other)

    trues = np.ones(len(digit))
    falses = np.full((len(other), 1), -1)
    classifications = np.append(trues, falses)

    digit = digit[:, [1, 2]]
    other = other[:, [1, 2]]
    data = np.append(digit, other, axis = 0)

    clf = svm.SVC(kernel = 'poly', C=C, degree = Q, coef0 = 1.0, gamma = 1)

    clf.fit(data, classifications)

    y_pred = clf.predict(data)
    comparisons = np.equal(y_pred, classifications)

    scores = cross_val_score(clf, data, classifications, cv = 10)
    errors = 1 - scores
    ECV += np.mean(errors)
print float(ECV) / runs
```

```
# Questions 9-10
from sklearn import svm
from sklearn import cross_validation
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
C = 10e6
Q = 2
target_digit = 1
vs_digit = 5
test = np.loadtxt("features.test.txt")
train = np.loadtxt("features.train.txt")

digit = []
digit2 = []
other = []
other2 = []
for row in train:
    if row[0] == target_digit:
        digit.append([row[0], row[1], row[2]])
    elif row[0] == vs_digit:
        other.append([row[0], row[1], row[2]])
for row in test:
    if row[0] == target_digit:
        digit2.append([row[0], row[1], row[2]])
    elif row[0] == vs_digit:
        other2.append([row[0], row[1], row[2]])
digit = np.asarray(digit)
other = np.asarray(other)
digit2 = np.asarray(digit2)
other2 = np.asarray(other2)

trues = np.ones(len(digit))
falses = np.full((len(other), 1), -1)
classifications = np.append(trues, falses)

trues2 = np.ones(len(digit2))
falses2 = np.full((len(other2), 1), -1)
classifications2 = np.append(trues2, falses2)

digit = digit[:, [1, 2]]
other = other[:, [1, 2]]
digit2 = digit2[:, [1, 2]]
other2 = other2[:, [1, 2]]
data = np.append(digit, other, axis = 0)
data2 = np.append(digit2, other2, axis = 0)

clf = svm.SVC(kernel = 'rbf', C=C, degree = Q, coef0 = 1.0, gamma = 1)

clf.fit(data, classifications)

y_pred = clf.predict(data)
y_pred2 = clf.predict(data2)

comparisons = np.equal(y_pred, classifications)
comparisons2 = np.equal(y_pred2, classifications2)
print 1 - float(np.sum(comparisons)) / len(comparisons)
print 1 - float(np.sum(comparisons2)) / len(comparisons2)
print len(clf.support_vectors_)
```