In [16]:

```
from sklearn.datasets import load_digits
```

In [29]:

```
digits = load_digits()
X = digits.data
y = digits.target

for i in range(len(y)):
    if (y[i] == 3):
        y[i] = 1
    else:
        y[i] = 0
```

In [30]:

```
from sklearn import neighbors
from sklearn.model_selection import train_test_split
import random

l = [i for i in range(len(X))]
random.shuffle(l)
X_train = []
y_train = []

for i in l:
    X_train.append(X[i])
    y_train.append(y[i])

X_test = X_train[-1300:]
X_train = X_train[:-1300]
y_test = y_train[-1300:]
y_train = y_train[:-1300]

clf = neighbors.KNeighborsClassifier(5, p=3)

clf.fit(X_train, y_train)

pred = clf.predict(X_test)
```

In [31]:

```
from sklearn.metrics import accuracy_score

print (accuracy_score(pred, y_test))
```

```
0.989230769231
```

In [32]:

```
tp = 0
tn = 0
fn = 0
fp = 0

for i in range(len(y_test)):
    if (y_test[i] == pred[i]):
        if pred[i] == 1:
            tp += 1
        else:
            tn += 1
    else:
        if (y_test[i] == 1):
            fn += 1
        else:
            fp += 1

tpr = tp/(tp+fn)
recall = tpr
fnr = fn/(tp+fn)
precision = tp/(tp+fp)
specificity = tn/(tn+fp)
sensitivity = tp/(tp+fn)
accuracy = (tp+tn)/(tp+tn+fn+fp)

print (tpr)
print (fnr)
print (precision)
print (recall)
print (accuracy)
print (specificity)
```

```
0.9140625
0.0859375
0.975
0.9140625
0.9892307692307692
0.9974402730375427
```

In [33]:

```
from sklearn.datasets import load_diabetes

data1 = load_diabetes()
X = data1.data
y = data1.target
```

In [34]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state =
 5)
```

In [36]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

theta,residuals,rank,s = np.linalg.lstsq(X_train, y_train)
predictions = np.dot(X_test, theta)
print (mean_squared_error(y_test, predictions))
print (mean_absolute_error(y_test, predictions))
```

```
28555.0214944
158.520100415
```

In [38]:

```
import numpy

def report_ablation_mse(X, y, X_test, y_test):
    mse_list = []
    for i in range(len(X[0])-1):
        X1 = numpy.delete(X, i+1, 1)
        theta,residuals,rank,s = numpy.linalg.lstsq(X1, y)
        mse = mean_squared_error(y_test, numpy.dot(numpy.delete(X_test,
i+1, 1), theta))
        mse_list.append(mse)
        print (mse)
    print (mse_list.index(min(mse_list)))
    print (mse_list.index(max(mse_list)))

report_ablation_mse(X, y, X_test, y_test)
```

```
26390.9220715
27393.6472803
26434.5802128
26644.2591277
26614.5913856
26589.8639416
26605.3305118
26866.7980856
26708.7614129
0
1
```