Univariate Data:

I Data Generation:

X = random('Normal',60,8,1,1000);

II MLE:

1 Get mean and Standard Deviation

```
>> m = sum(X)/length(X)

m =

   60.2952

>> sigma = sqrt(sum((X-m).^2)/length(X))

sigma =

    7.9848
```

2 MLE function also returns the 95% confidence intervals for the parameters

```
>> [T,pci] = mle(X)

T =

   60.2952    7.9848


pci =

   59.7994    7.6533
   60.7909    8.3552
```

3 The values got from MLE function are as same as above. Both of these two methods are using same equation to get mean and standard deviation.

III MAP and Bayes' Estimator: Assume that collection of all these possible parameter value estimates is also distributed normally. That is, $X \sim N(\theta,\sigma^2)$ and $\theta \sim N(\mu_0,\sigma_0^2)$. Assume that σ=8, μ_0=60, σ_0=3.

1 In this case of normal density and $p(\theta|X)$ is normal, then $\theta_{MAP} = \theta_{Bayes'}$

The MAP estimate and the Bayes' estimate the same in this case.

$\theta_{MAP} = \theta_{Bayes'} =$

$$E[\theta \mid \mathcal{X}] = \frac{N/\sigma_0^2}{N/\sigma_0^2 + 1/\sigma^2} m + \frac{1/\sigma^2}{N/\sigma_0^2 + 1/\sigma^2} \mu$$

```
>> N = 1000; sigma=8; sigma0=3; u=60;

>> map = N/sigma0/(N/sigma0^2+1/sigma^2)*m +
1/sigma^2/(N/sigma0^2+1/sigma^2)*u

map =

  180.8686
```

2 If we have no prior reason to favor some values of θ, then the prior density is flat and the posterior will have the same form as the likelihood, p(X| θ), and the MAP estimate will be equivalent to the maximum likelihood estimate. In this case, we have prior reason, $\theta \sim N(\mu_0,\sigma_0^2)$, then the MLE is not equal to MAP.

IV **Classification:**

1
```
C1 = random('Normal', 60,8,1,500);
C2 = random('Normal', 30,12,1,300);
C3 = random('Normal', 80,4,1,200);
X(1:500,1) = C1;X(1:500,2) = 1;X(1:500,3) = 0;X(1:500,4) = 0;
X(501:800,1) = C2;X(501:800,2) = 0;X(501:800,3) = 1;X(501:800,4) = 0;
X(801:1000,1) = C3;X(801:1000,2) = 0;X(801:1000,3) = 0;X(801:1000,4) = 1;
```

2 Discriminant Function

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log s_i - \frac{(x-m_i)^2}{2s_i^2} + \log \hat{P}(C_i)$$

Apply MLE to estimate the parameters of each of the classes:

$$m_i = \frac{\sum_t x^t r_i^t}{\sum_t r_i^t}$$

$$s_i^2 = \frac{\sum_t (x^t - m_i)^2 r_i^t}{\sum_t r_i^t}$$

```
m1=sum(X(:,1).*X(:,2))/sum(X(:,2))
m1 = 60.0270
m2=sum(X(:,1).*X(:,3))/sum(X(:,3))
m2 = 29.2232
m3=sum(X(:,1).*X(:,4))/sum(X(:,4))
```

m3 = 80.2365
s1 = sqrt(sum((X(:,1)-m1).^2.*X(:,2))/sum(X(:,2)))
s1 = 8.3139
s2 = sqrt(sum((X(:,1)-m2).^2.*X(:,3))/sum(X(:,3)))
s2 = 11.3131
s3 = sqrt(sum((X(:,1)-m3).^2.*X(:,4))/sum(X(:,4)))
s3 = 3.9514
P1_hat = sum(X(:,2))/length(X) = 0.5
P2_hat = sum(X(:,3))/length(X) = 0.3
P3_hat = sum(X(:,4))/length(X) = 0.2

3 Choosing Class based on the inputs:
The discriminant function is:
g_i(x) = - log s_i - ((x - m_i)^2/(2 s_i^2)) + logP(C_i)_hat
The mean vector is:
m = [m1,m2,m3]
The standard deviation vector is:
s = [s1,s2,s3]
The prior probability vector is:
P_hat = [P1_hat, P2_hat, P3_hat]

Define a Classify Function:

```
function [ g ] = classify( x, M, S, P_hat )
% Calculate the discriminant function to classify the input
% x based on the mean, standard deviation, and prior
% probability getting from MLE
    for i = 1: length(M)
       m = M(i);s = S(i); p_hat = P_hat(i);
       g = -log(s) - ((x - m)^2/(2*s^2)) + log(p_hat);
       D(i) = g;
    end
    G = max(D);
    for i = 1:length(D)
       if G == D(i)
           g = i;
       end
    end
end
```

```
>> classify(10, m, s, P_hat)
ans = 2
>> classify(30, m, s, P_hat)
ans = 2
>> classify(50, m, s, P_hat)
```

ans = 1
>> classify(70, m, s, P_hat)
ans = 1
>> classify(90, m, s, P_hat)
ans = 3

4. Because the discriminant function is

$$g_i(x) = \log p(x|C_i) + \log P(C_i)$$

$$p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right]$$

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log s_i - \frac{(x-m_i)^2}{2s_i^2} + \log \hat{P}(C_i)$$

The mean of each class is : m1 = 60.0907; m2 = 29.3310; m3 = 79.8486;
The analytical "decision thresholds" is $g_1(x) = g_2(x)$, and $g_1(x) = g_3(x)$

5&6. Discriminant function:
Define a Classify Function:

```
function [ g ] = classify( x, M, S, P_hat )
% Calculate the discriminant function to classify the input
% x based on the mean, standard deviation, and prior
% probability getting from MLE
    for i = 1: length(M)
       m = M(i);s = S(i); p_hat = P_hat(i);
       g = -log(s) - ((x - m)^2/(2*s^2)) + log(p_hat);
       D(i) = g;
    end
    G = max(D);
    for i = 1:length(D)
       if G == D(i)
           g = i;
       end
    end
end
```

Input: x, mean vector, std vector, p_hat vector
output: class number
For each class distribution, calculate the discriminant function, then get the maximum discriminant function and return the class number.

7. inputs: x = 0, 0.5, 1, 1.5, ..., 99, 99.5, 100
The "decision thresholds" calculated analytically will coincide with the results of this test. The reason is we can use the every x as an input and calculate the discriminant

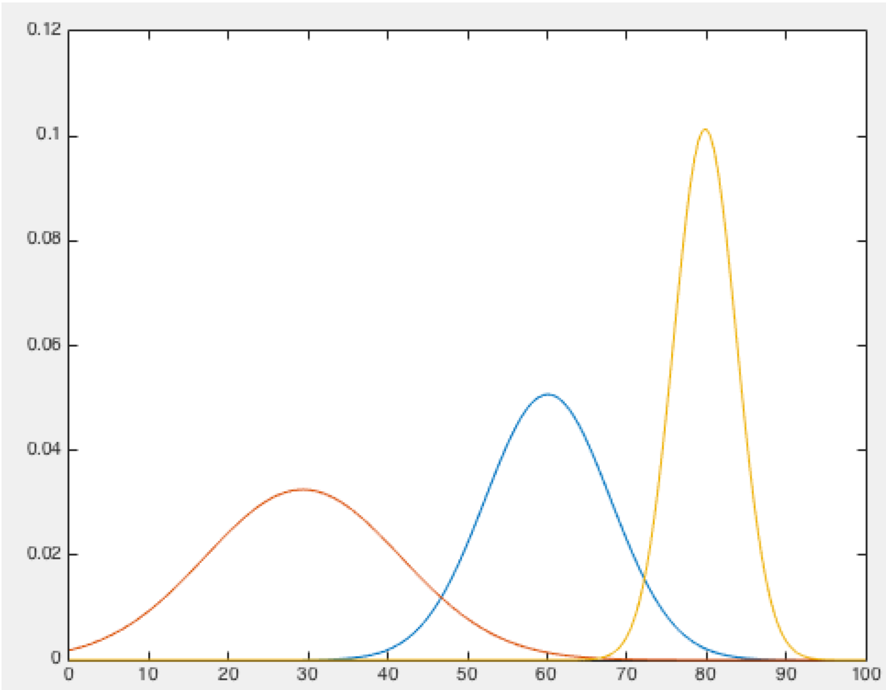
function and choose the largest one as the class that x belonging to.

8. >> xval = 0:0.5:100;

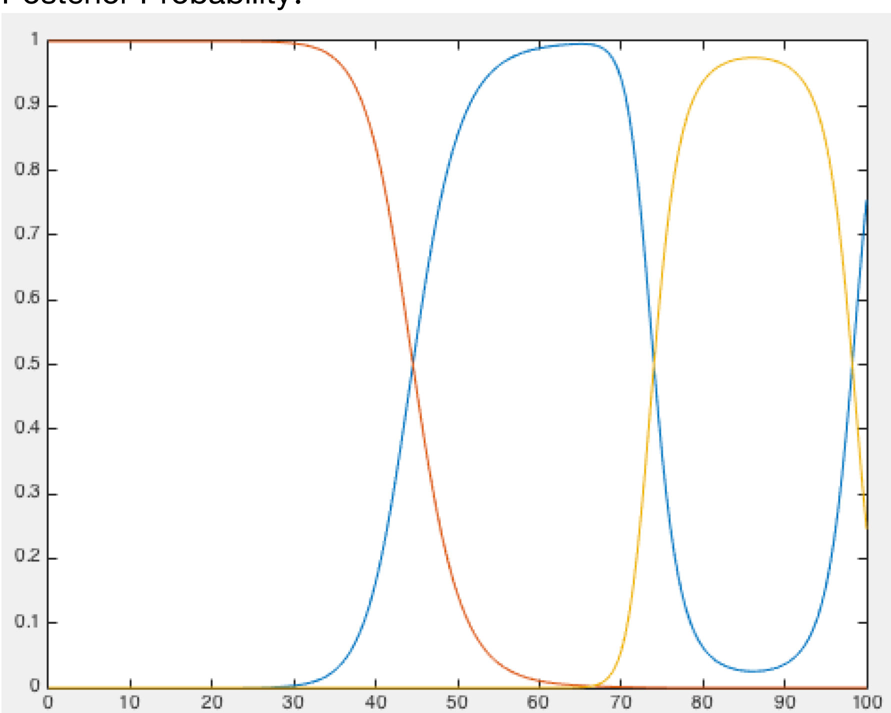
```
function plot_likelihood (xval, m, s )
    xval = 0:0.5:100;
    Y = zeros(length(xval),length(xval),length(xval));
    for i = 1:3
        Y(:,i) = normpdf(xval,m(i),s(i));
    end
    plot(xval, Y(:,1),xval, Y(:,2),xval, Y(:,3));
end

function plot_posterior(xval , m, s, P_hat)
    yval = zeros(length(xval),length(xval),length(xval));
    for i = 1:length(xval)
        px = 0.0;
        for j = 1:length(m)
            px = px + normpdf(xval(i),m(j),s(j))*P_hat(j);
        end
        for j = 1:length(m)
           yval(i,j) =
normpdf(xval(i),m(j),s(j))*P_hat(j)/px;
        end
    end
    plot(xval, yval(:,1),xval, yval(:,2),xval, yval(:,3));
end
```

```
>> plot_class(xval, m, s)

>> plot_posterior(xval , m, s, P_hat)
```

Likelihood Probability:



Posterior Probability：



From the graph we can see that, x will be classified to class 2 when x less than 45, be classified to class 1 when x between 45 and 75, and be classified to class 3 when x greater than 75.

9. Random Sample:

```
>> [Train, Test] = crossvalind('HoldOut', length(X), 0.4);
>> training = X(Train,:);
>> validation = X(Test, :);
```

```
function [crossmatrix,precision] = validateClass(validation, m,
s, P_hat)
    crossmatrix = zeros(3,3);
    precision = [0,0,0];
    for i = 1:length(validation)
       x = validation(i,1);
       c_predict = classify(x, m, s, P_hat);
       c_actual = 0;
       for j = 2:4
           if validation(i,j) == 1
               c_actual = j-1
           end
       end
       crossmatrix(c_predict, c_actual) = crossmatrix(c_predict,
c_actual) + 1;
    end

    for i = 1:3
        precision(i) = crossmatrix(i,i)/ sum(crossmatrix(i,:));
    end

end
```

```
>> [crossmatrix,precision] = validateClass(validation, m, s, P_hat)

crossmatrix =

  199    10     4
    3   113     0
    4     0    67


precision =

  0.9343    0.9741    0.9437
```

V Regression:

1. $r = f(x) + \varepsilon$ where $f(x) = 2*\sin(1.5*x)$, and the noise $\varepsilon \sim N(\mu=0,\sigma^2=1)$.

```
>> e = random('Normal',0,1,1,1000);
```

```
>> r = sin(1.5*X(:,1))*2;
```

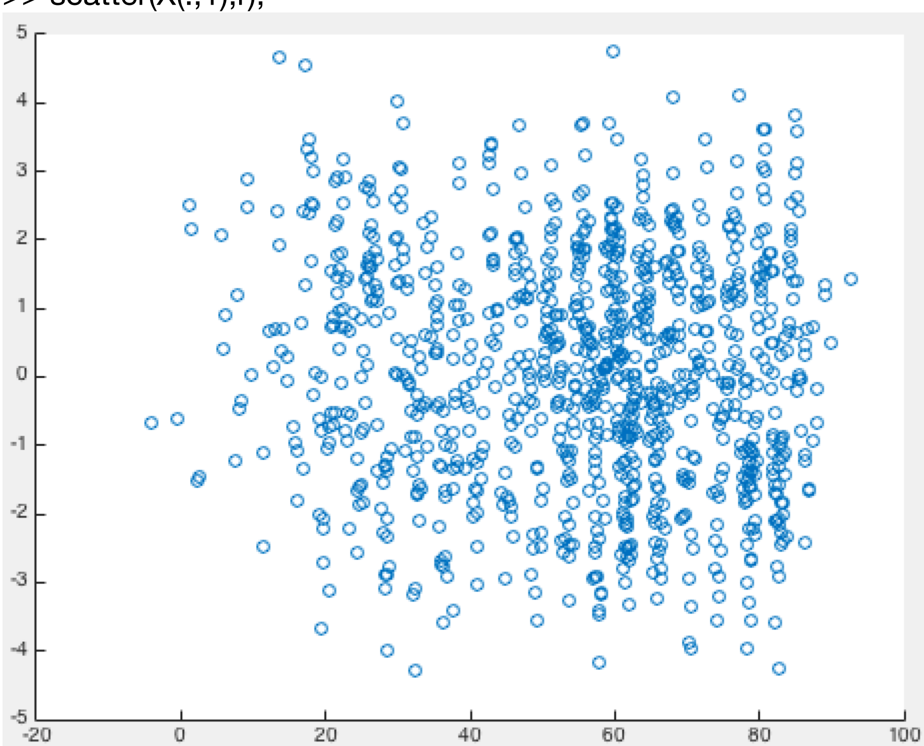
2. split your dataset

```
>> [Train, Test] = crossvalind('HoldOut', length(X), 0.4);

>> X_new = X(:,1);
>> train_x = X_new(Train,:);
>> val_x = X_new(Test, :);
>> train_e = e(Train);
>> val_e = e(Test);
>> train_y = sin(1.5*train_x)*2+train_e';
>> val_y = sin(1.5*val_x)*2+val_e';
```

3. Create three 2-dimensional plots:

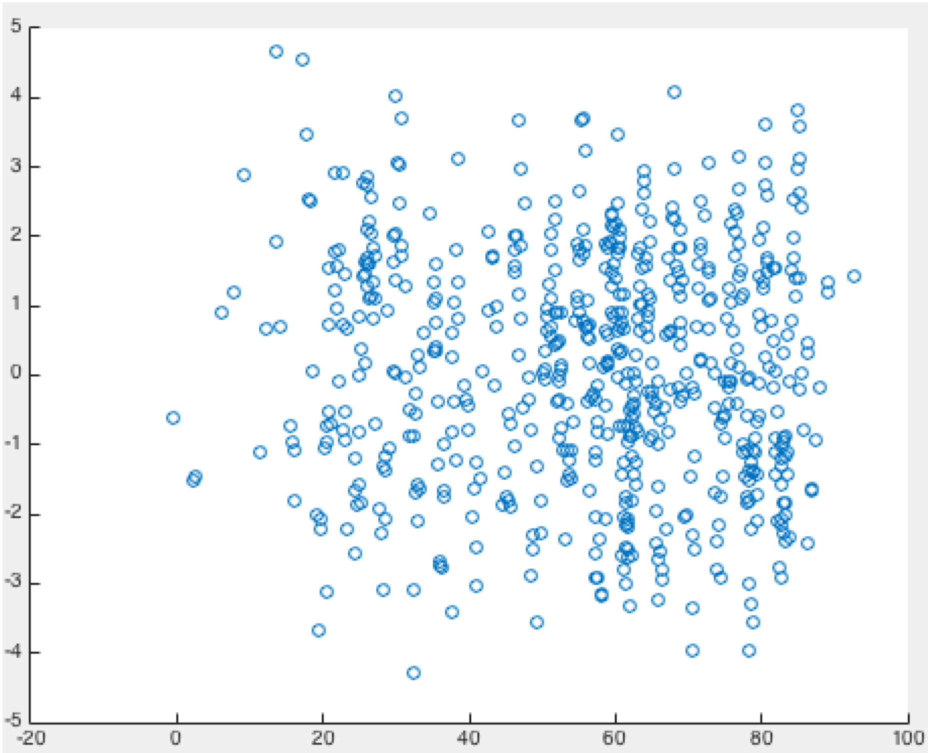
the entire dataset X：

```
>> scatter(X(:,1),r);
```

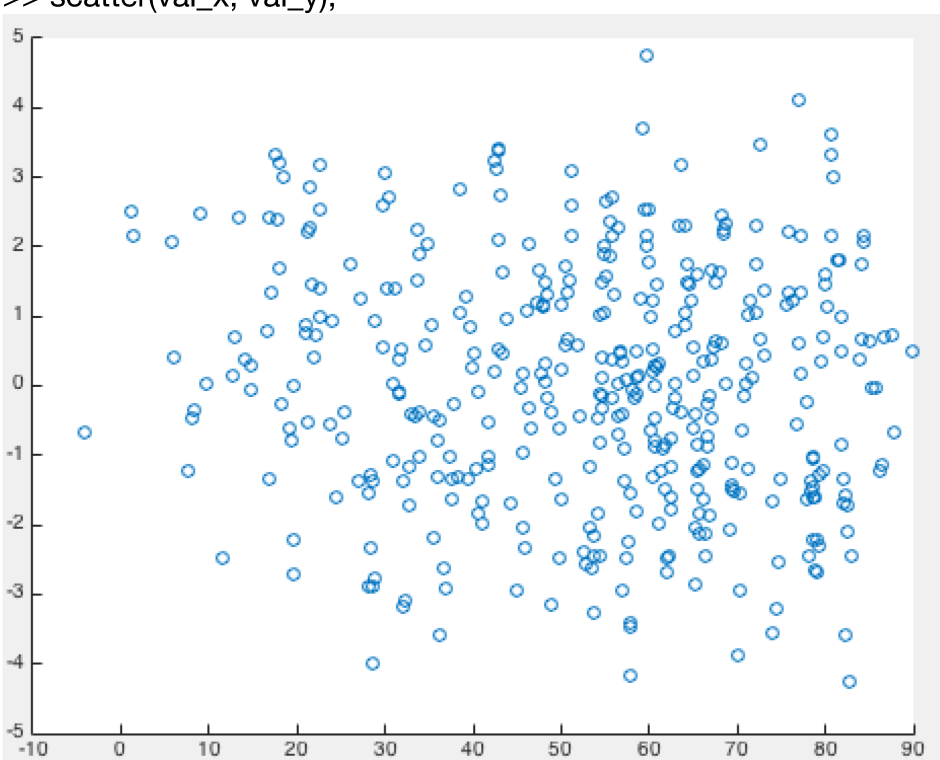


the training set：

```
>> scatter(train_x, train_y);
```



the validation set：
>> scatter(val_x, val_y);

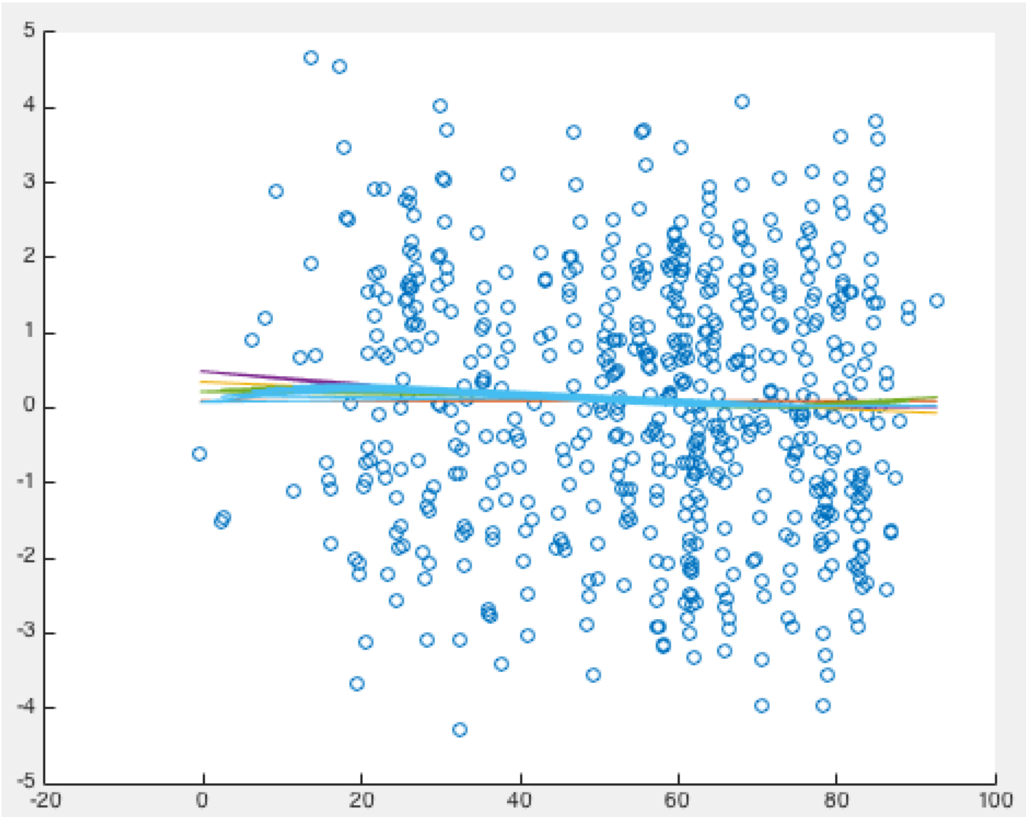


4: Regressions:

```
>> p0 = polyfit(train_x, train_y, 0);
>> p1 = polyfit(train_x, train_y, 1);
>> p2 = polyfit(train_x, train_y, 2);
>> p3 = polyfit(train_x, train_y, 3);
>> p4 = polyfit(train_x, train_y, 4);

>> y0_hat = polyval(p0, train_x);
>> y1_hat = polyval(p1, train_x);
>> y2_hat = polyval(p2, train_x);
>> y3_hat = polyval(p3, train_x);
>> y4_hat = polyval(p4, train_x);
```

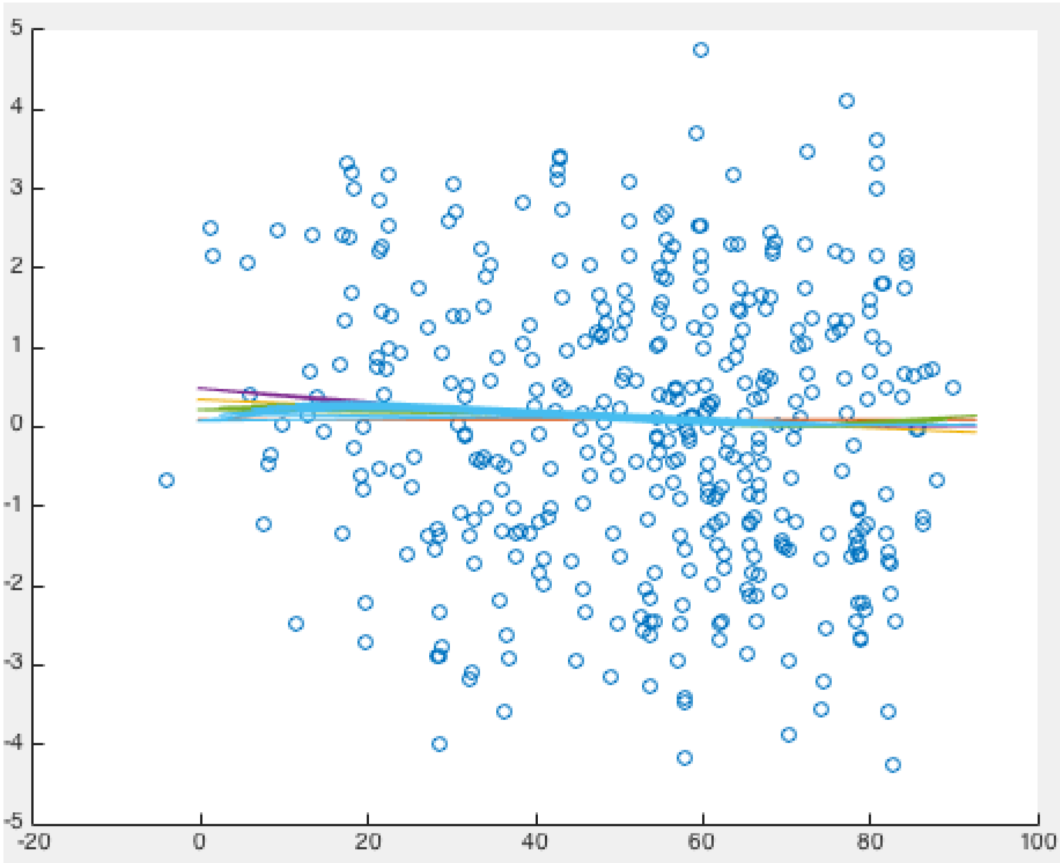
Training Data:

```
>> figure
>> scatter(train_x, train_y)
>> hold on
>> plot(train_x, y0_hat)
>> plot(train_x, y1_hat)
>> plot(train_x, y2_hat)
>> plot(train_x, y3_hat)
>> plot(train_x, y4_hat)
>> hold off
```



Validation Data:

```
>> scatter(val_x, val_y)
>> hold on
>> plot(train_x, y0_hat)
>> plot(train_x, y1_hat)
>> plot(train_x, y2_hat)
>> plot(train_x, y3_hat)
>> plot(train_x, y4_hat)
>> hold off
```



6. Model Selection:

Linear regression model:
    train_y ~ 1

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	0.096315	0.069623	1.3834	0.16706

Number of observations: 600, Error degrees of freedom: 599
Root Mean Squared Error: 1.71

Linear regression model:
    train_y ~ 1 + train_x

Estimated Coefficients:

Estimate	SE	tStat	pValue

```
    (Intercept)      0.34626      0.20468     1.6917    0.091218
    train_x       -0.0044575    0.0034328    -1.2985     0.19462


Number of observations: 600, Error degrees of freedom: 598
Root Mean Squared Error: 1.7
R-squared: 0.00281,  Adjusted R-Squared 0.00114

Linear regression model:
    train_y ~ 1 + train_x + train_x^2

Estimated Coefficients:
                   Estimate        SE         tStat      pValue
                  __________    __________    ________    _______

    (Intercept)      0.48565       0.41753      1.1632    0.24523
    train_x        -0.010947      0.017283    -0.63339    0.52672
    train_x^2     6.3143e-05    0.00016482     0.38311    0.70177


Number of observations: 600, Error degrees of freedom: 597
Root Mean Squared Error: 1.71
R-squared: 0.00306,  Adjusted R-Squared -0.000283

Linear regression model:
    train_y ~ 1 + train_x + train_x^2 + train_x^3

Estimated Coefficients:
                    Estimate        SE         tStat      pValue
                  ___________    __________    ________    _______

    (Intercept)       0.22319      0.72308     0.30867    0.75768
    train_x          0.010569      0.05138     0.20571    0.83709
    train_x^2     -0.00041973    0.0010982    -0.38218    0.70247
    train_x^3      3.1983e-06   7.1918e-06     0.44472    0.65669


Number of observations: 600, Error degrees of freedom: 596
Root Mean Squared Error: 1.71
R-squared: 0.00339,  Adjusted R-Squared -0.00163

Linear regression model:
    train_y ~ 1 + train_x + train_x^2 + train_x^3 + train_x^4
```

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	0.091497	0.92136	0.099306	0.92093
train_x	0.029983	0.098544	0.30426	0.76104
train_x^2	-0.0012452	0.0037397	-0.33297	0.73928
train_x^3	1.6353e-05	5.7416e-05	0.28481	0.77589
train_x^4	-6.9946e-08	3.0289e-07	-0.23093	0.81745

Number of observations: 600, Error degrees of freedom: 595
Root Mean Squared Error: 1.71
R-squared: 0.00348, Adjusted R-Squared -0.00322

Based on these error measures, I will choose the model: train_y ~ 1. The reason is that all of the five models are bad and the Root Mean Squared Error are same. Then we should choose the simplest model we have.

Multivariate Data:

I. Multivariate Normal Distribution:

1) Multivariate Data Generation:

```
>> rng(100)
>> X1 = mvnrnd(trueMeans,trueSigmaA,1000);
>> X2 = mvnrnd(trueMeans,trueSigmaD,1000);
>> X3 = mvnrnd(trueMeans, eye(20,20),1000);
```

2) Parameter Estimation:

```
function [sampleMean, sampleVar] = estimateMultivariate(sample)
    sampleMean = zeros(1,20);
    sampleVar = zeros(20,20);
    N = length(sample);
    for i = 1:20
       sampleMean(i) = sum(sample(:,i))/N;
    end
    for i = 1:20
       for j = 1:20
          sampleVar(i,j) = sum((sample(:,i) -
sampleMean(i)).*(sample(:,j) - sampleMean(j)))/N;
       end
```

```
    end
end
```

```
>> [sampleMean1, sampleVar1] = estimateMultivariate(X1);

>> [sampleMean2, sampleVar2] = estimateMultivariate(X2);

>> [sampleMean3, sampleVar3] = estimateMultivariate(X3);
```

From the result we can see that the sampleMean and sampleVarariance are close to the trueMeans and trueVarairance.

II Multivariate Classification:

1) Multivariate Data Generation:

a) Dataset DX (*classes have different arbitrary covariance matrices)*:

```
>> X12 = mvnrnd(trueMeans2, trueSigmaA2, 800);
>> DX(1:1000,:) = X1;
>> DX(1001:1800,:) = X12;

>> DX(1:1000,21) = 1;
>> DX(1001:1800,21) = 0;
>> DX(1:1000,22) = 0;
>> DX(1001:1800,22) = 1;
```

b) Dataset SX1*(classes share the same arbitrary covariance matrix):*

```
>> SX12 = mvnrnd(trueMeans2, trueSigmaA, 800);
>> SX1(1:1000,:) = X1;
>> SX1(1001:1800,:) = SX12;

>> SX1(1:1000,21) = 1;
>> SX1(1001:1800,21) = 0;
>> SX1(1:1000,22) = 0;
>> SX1(1001:1800,22) = 1;
```

c) Dataset SX2 *(classes share the same diagonal covariance matrix):*

```
>> SX22 = mvnrnd(trueMeans2, trueSigmaD, 800);
>> SX2(1:1000,:) = X2;
>> SX2(1001:1800,:) = SX22;
```

```
>> SX2(1:1000,21) = 1;
>> SX2(1001:1800,21) = 0;
>> SX2(1:1000,22) = 0;
>> SX2(1001:1800,22) = 1;
```

d) Dataset SX3 (*classes share the identity covariance matrix)*:

```
>> SX31 = mvnrnd(trueMeans2, eye(20,20), 800);
>> SX3(1:1000,:) = X3;
>> SX3(1001:1800,:) = SX31;

>> SX3(1:1000,21) = 1;
>> SX3(1001:1800,21) = 0;
>> SX3(1:1000,22) = 0;
>> SX3(1001:1800,22) = 1;
```

2) Multivariate Discriminant Functions:

The discriminant function g_i for each class C_i of the dataset should be:
First use the MLE to calculate the parameters, including prior probability, sample means, and sample variances.

$$
\begin{aligned}
\hat{P}(C_i) &= \frac{\sum_t r_i^t}{N} \\
\boldsymbol{m}_i &= \frac{\sum_t r_i^t \boldsymbol{x}^t}{\sum_t r_i^t} \\
\mathbf{S}_i &= \frac{\sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T}{\sum_t r_i^t}
\end{aligned}
$$

Then we get the discriminant function as:

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^T \mathbf{S}_i^{-1}(\boldsymbol{x} - \boldsymbol{m}_i) + \log \hat{P}(C_i)$$

We will calculate each g_i for the given record, and select the class C_i to maximize g_i as the record class.

Implement each of these 2 discriminant function g_i:

```
% Classify Multivariant Variable:
function [ c ] = multivariantClassify(x, sample)
    c = 0;
    C1 = sample(1:1000,1:20);
    C2 = sample(1001:1800,1:20);
    N1 = length(C1);
```

```
    N2 = length(C2);
    p1_hat = N1/(N1+N2);
    p2_hat = N2/(N1+N2);
    [m1, s1] = estimateMultivariate(C1);
    [m2, s2] = estimateMultivariate(C2);

    g(1) = multivarDiscriminant(x,m1,s1,p1_hat);
    g(2) = multivarDiscriminant(x,m2,s2,p2_hat);

    if g(1) > g(2)
        c = 1;
    else
        c = 2;
    end
end


% Discriminant Function:
function g = multivarDiscriminant(x,m,s,p_hat)
    g = -1/2*log(det(s)) - 1/2*(x-m)*inv(s)*(x-m)' + log(p_hat)
end


% Validate the Multi-Class:
function [crossmatrix,accuracy] = validateMultiClass(x,all_samples)
    crossmatrix = zeros(2,2);
    accuracy = [0,0];
    for row = 1:length(x)
        c_predict = multivariantClassify(x(row,1:20),all_samples);
        for col = 21:22
           if x(row,col) == 1
               c_actual = col - 20;
           end
        end
        crossmatrix(c_predict, c_actual) = crossmatrix(c_predict,
c_actual) + 1;
    end

    for i = 1:length(accuracy)
        accuracy(i) = crossmatrix(i,i)/ sum(crossmatrix(i,:));
    end
end
```

1) DX dataset:

```
>> x = DX(1,1:20);
```

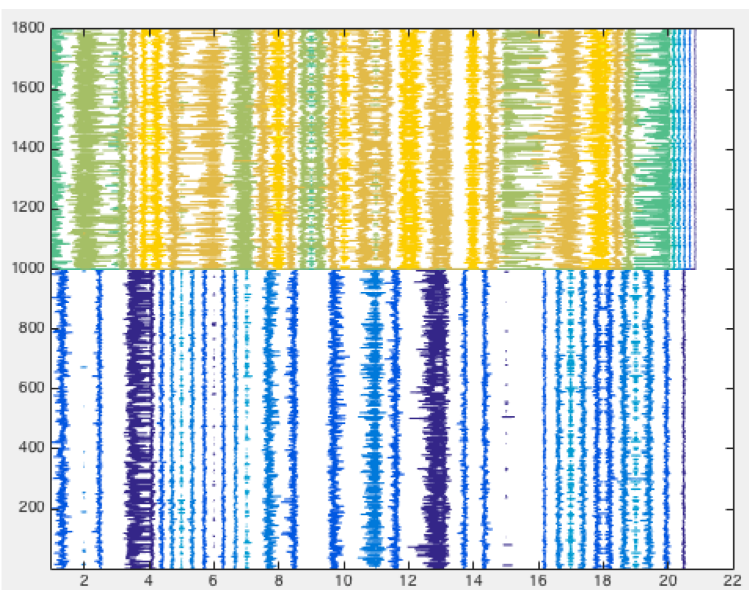
```
>> [ c ] = multivariantClassify(x, DX)
c =

     1
```

This point x is classified to 1.

Report the accuracy and the confusion matrix

```
>> [Train, Test] = crossvalind('HoldOut', length(DX), 0.4);
>> training = DX(Train,:);
>> validation = DX(Test, :);

>> [crossmatrix,accuracy] = validateMultiClass(validation ,DX);
crossmatrix =
      396     0
        0   324

   accuracy =
           1     1
```

DX Data Visualization:



2**) SX1 Dataset**

```
>> x = SX1(1,1:20);

>> [ c ] = multivariantClassify(x, SX1)
c =
```

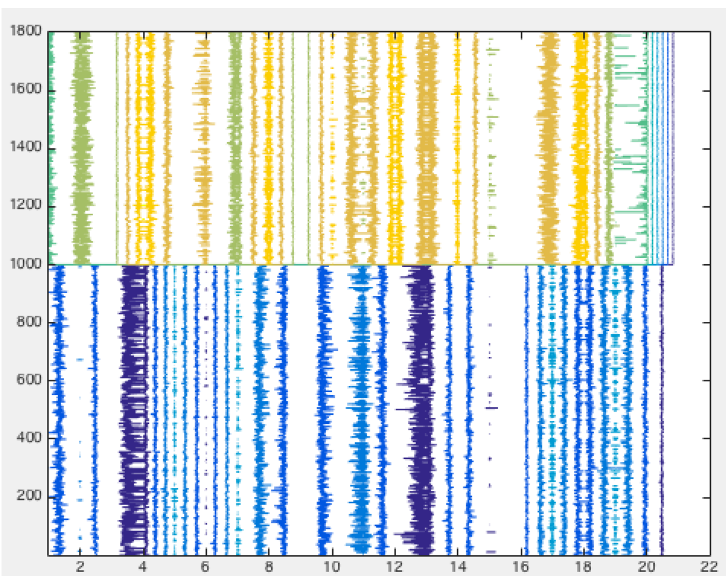
1

This point x is classified to 1.

Report the accuracy and the confusion matrix

```
>> [Train, Test] = crossvalind('HoldOut', length(SX1), 0.4);
>> training = SX1 (Train,:);
>> validation = SX1 (Test, :);

>> [crossmatrix,accuracy] = validateMultiClass(validation , SX1);
   crossmatrix =

      382     0
        0   338


   accuracy =

        1     1
```

SX1 Dataset Visualization:



3**) SX2 Dataset**

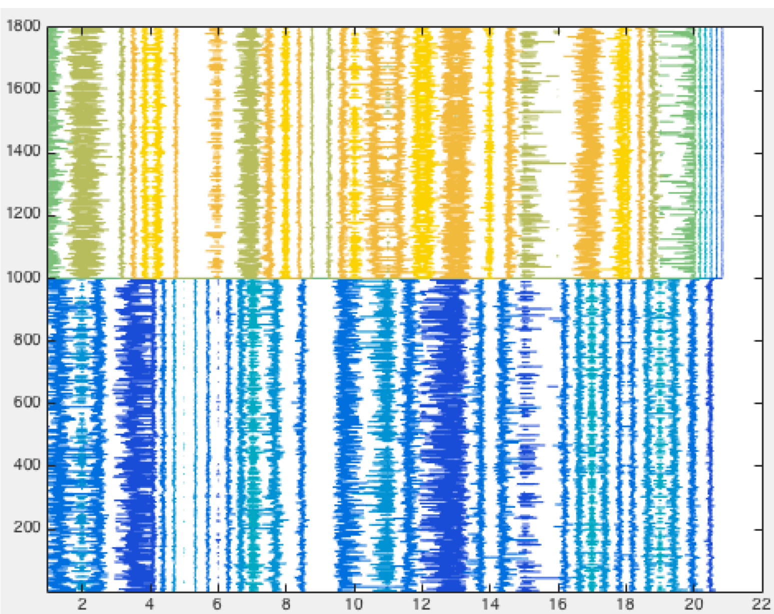
```
>> x = SX2(1,1:20);

>> [ c ] = multivariantClassify(x, SX2)
c =
```

1
This point x is classified to 1.

Report the accuracy and the confusion matrix

```
>> [Train, Test] = crossvalind('HoldOut', length(SX2), 0.4);
>> training = SX2 (Train,:);
>> validation = SX2 (Test, :);

>> [crossmatrix,accuracy] = validateMultiClass(validation , SX2);
   crossmatrix =

      417     0
        0   303


   accuracy =

        1     1
```

SX2 Dataset Visualization:



4**) SX3 Dataset**

```
>> x = SX3(1,1:20);

>> [ c ] = multivariantClassify(x, SX3)
```

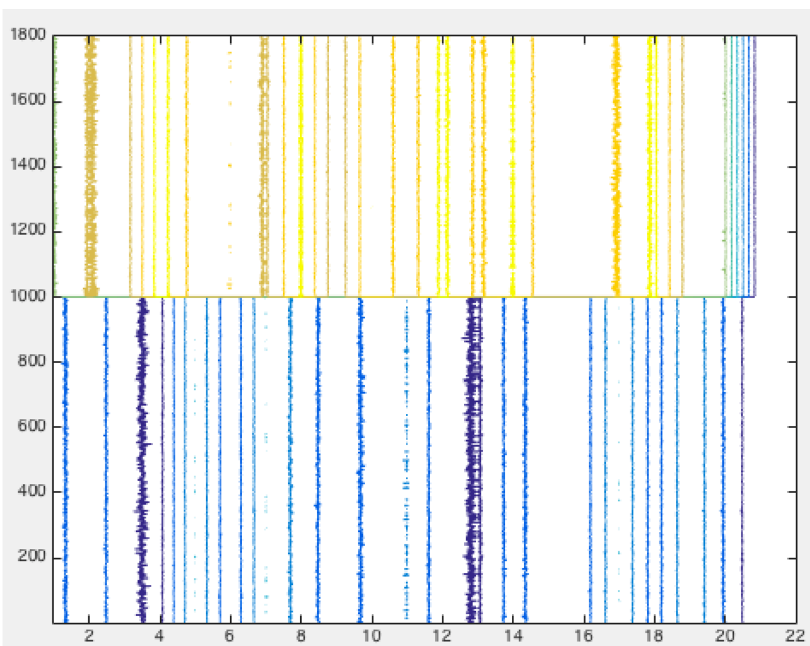
```
c =

     1
```

This point x is classified to 1.

Report the accuracy and the confusion matrix

```
>> [Train, Test] = crossvalind('HoldOut', length(SX3), 0.4);
>> training = SX3 (Train,:);
>> validation = SX3 (Test, :);

>> [crossmatrix,accuracy] = validateMultiClass(validation , SX3);

   crossmatrix =

     391     0
       0   329


   accuracy =

       1     1
```

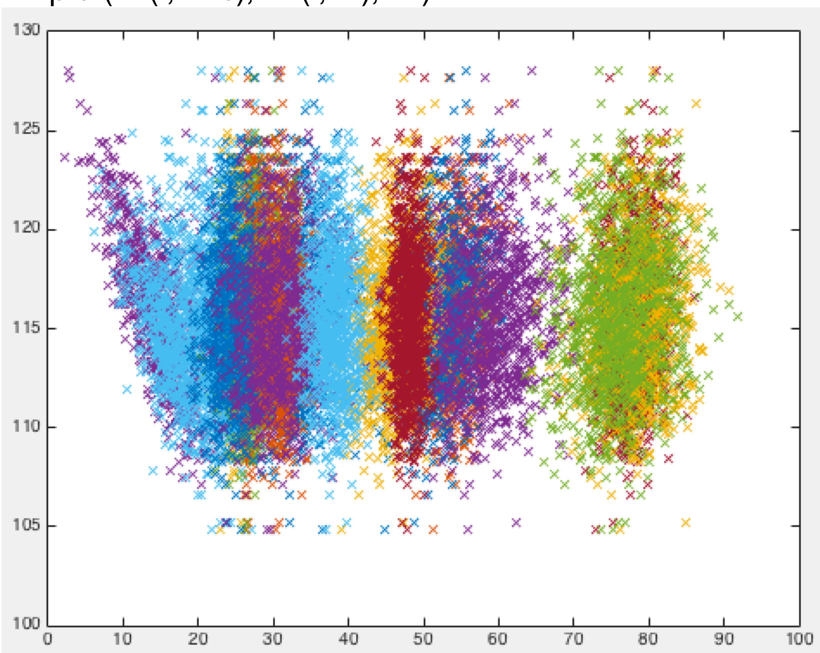
SX3 Dataset Visualization:



III Multivariate Regression:

1. >> e_mr = normrnd(0,1,1000,1);
>> output = 3*mean(X1,2)-min(X1')' + e_mr;
>> X1(:,1:21) = output;

>> plot(X1(:,1:20), X1(:,21), 'x')



2.
>> [Train, Test] = crossvalind('HoldOut', length(X1), 0.4);
>> training = X1(Train,:);
>> validation = X1(Test, :);

3. >> lm_model = fitlm(training(:,1:20),training(:,21))

```
Linear regression model:
    y ~ [Linear formula with 21 terms in 20 predictors]

Estimated Coefficients:
                   Estimate       SE         tStat        pValue  
                   ________    ________    ________    __________

    (Intercept)     -10.067      10.194    -0.98755       0.32378
    x1              0.23284    0.034265      6.7953     2.692e-11
    x2              0.18215    0.034626      5.2604    2.0264e-07
    x3              0.31219    0.089098      3.5039    0.00049397
    x4             -0.53195    0.023402     -22.731    3.0709e-82
    x5              0.16325    0.034484      4.7341    2.7702e-06
    x6              0.15979    0.033269       4.803    1.9932e-06
    x7               0.1221    0.042568      2.8683     0.0042777
    x8              0.17446    0.023031      7.5751    1.4285e-13
    x9              0.40927     0.21145      1.9355       0.05341
    x10             0.12528    0.031075      4.0314    6.2859e-05
    x11             0.15068    0.018612      8.0958    3.3688e-15
    x12             0.10719    0.040102       2.673     0.0077301
    x13            -0.14477    0.024182     -5.9867    3.7571e-09
    x14             0.17248    0.069069      2.4972      0.012796
    x15            0.072899    0.034969      2.0847      0.037537
    x16             0.17194    0.059936      2.8687     0.0042713
    x17             0.16855    0.024417       6.903    1.3414e-11
    x18             0.14606    0.023906      6.1096    1.8341e-09
    x19             0.15069    0.020443      7.3713    5.8675e-13
    x20            0.071411     0.04809      1.4849       0.13811


Number of observations: 600, Error degrees of freedom: 579
Root Mean Squared Error: 1.97
R-squared: 0.698,  Adjusted R-Squared 0.687
F-statistic vs. constant model: 66.9, p-value = 5.95e-136
```

4. >> SSE = sum((predict(:,21)-predict(:,22)).^2);
SSE =

  1.2583e+03

>> y_mean = mean(predict(:,21))

y_mean =

```
  115.5272

>> SSR = sum((predict(:,22)-y_mean).^2)

SSR =

   3.5694e+03

>> SST = sum((predict(:,21) - y_mean).^2)

SST =

   4.7033e+03

>> R_square = SSR/SST

R_square =

    0.7589


>> MSE = SSE/400

MSE =

    3.1458

>> RMSE = MSE^(1/2)

RMSE =

    1.7736

>> RSE = sum((predict(:,22)-predict(:,21)).^2)/sum((predict(:,21)-y_mean).^2)

RSE =

    0.2675
```

5.

6.

```
>> D1 = mvnrnd(trueMeans,trueSigmaA,100);
```

```
>> D2 = mvnrnd(trueMeans,trueSigmaA,100);
>> D3 = mvnrnd(trueMeans,trueSigmaA,100);
>> D4 = mvnrnd(trueMeans,trueSigmaA,100);
>> D5 = mvnrnd(trueMeans,trueSigmaA,100);
>> D6 = mvnrnd(trueMeans,trueSigmaA,100);
>> D7 = mvnrnd(trueMeans,trueSigmaA,100);
>> D8 = mvnrnd(trueMeans,trueSigmaA,100);
>> D9 = mvnrnd(trueMeans,trueSigmaA,100);
>> D10 = mvnrnd(trueMeans,trueSigmaA,100);
>>
>> Y1 = 3*mean(D1,2) - min(D1')' + normrnd(0,1,100,1);
>> Y2 = 3*mean(D2,2) - min(D2')' + normrnd(0,1,100,1);
>> Y3 = 3*mean(D3,2) - min(D3')' + normrnd(0,1,100,1);
>> Y4 = 3*mean(D4,2) - min(D4')' + normrnd(0,1,100,1);
>> Y5 = 3*mean(D5,2) - min(D5')' + normrnd(0,1,100,1);
>> Y6 = 3*mean(D6,2) - min(D6')' + normrnd(0,1,100,1);
>> Y7 = 3*mean(D7,2) - min(D7')' + normrnd(0,1,100,1);
>> Y8 = 3*mean(D8,2) - min(D8')' + normrnd(0,1,100,1);
>> Y9 = 3*mean(D9,2) - min(D9')' + normrnd(0,1,100,1);
>> Y10 = 3*mean(D10,2) - min(D10')' + normrnd(0,1,100,1);

>> g1 = fitlm(D1,Y1);
>> g2 = fitlm(D2,Y2);
>> g3 = fitlm(D3,Y3);
>> g4 = fitlm(D4,Y4);
>> g5 = fitlm(D5,Y5);
>> g6 = fitlm(D6,Y6);
>> g7 = fitlm(D7,Y7);
>> g8 = fitlm(D8,Y8);
>> g9 = fitlm(D9,Y9);
>> g10 = fitlm(D10,Y10);

>> f1 = 3*mean(D1,2) - min(D1')';
>> f2 = 3*mean(D2,2) - min(D2')';
>> f3 = 3*mean(D3,2) - min(D3')';
>> f4 = 3*mean(D4,2) - min(D4')';
>> f5 = 3*mean(D5,2) - min(D5')';
>> f6 = 3*mean(D6,2) - min(D6')';
>> f7 = 3*mean(D7,2) - min(D7')';
>> f8 = 3*mean(D8,2) - min(D8')';
>> f9 = 3*mean(D9,2) - min(D9')';
>> f10 = 3*mean(D10,2) - min(D10')';

>> Y1_hat = g1.predict(D1);
>> Y2_hat = g2.predict(D2);
```

```
>> Y3_hat = g3.predict(D3);
>> Y4_hat = g4.predict(D4);
>> Y5_hat = g5.predict(D5);
>> Y6_hat = g6.predict(D6);
>> Y7_hat = g7.predict(D7);
>> Y8_hat = g8.predict(D8);
>> Y9_hat = g9.predict(D9);
>> Y10_hat = g10.predict(D10);
```

Bias:

```
>> SB1 = sum((mean(Y1_hat) - f1).^2)/100;
>> SB2 = sum((mean(Y2_hat) - f2).^2)/100;
>> SB3 = sum((mean(Y3_hat) - f3).^2)/100;
>> SB4 = sum((mean(Y4_hat) - f4).^2)/100;
>> SB5 = sum((mean(Y5_hat) - f5).^2)/100;
>> SB6 = sum((mean(Y6_hat) - f6).^2)/100;
>> SB7 = sum((mean(Y7_hat) - f7).^2)/100;
>> SB8 = sum((mean(Y8_hat) - f8).^2)/100;
>> SB9 = sum((mean(Y9_hat) - f9).^2)/100;
>> SB10 = sum((mean(Y10_hat) - f10).^2)/100;
SB1 =10.9543
SB2 =12.5871
SB3 =11.4534
SB4 =11.2504
SB5 =8.5286
SB6 =10.2179
SB7 =9.7556
SB8 =10.8861
SB9 =12.9570
SB10 =9.1709
```

Variance:

```
>> VAR1 = sum((mean(Y1_hat) - Y1).^2)/100;
>> VAR2 = sum((mean(Y2_hat) - Y2).^2)/100;
>> VAR3 = sum((mean(Y3_hat) - Y3).^2)/100;
>> VAR4 = sum((mean(Y4_hat) - Y4).^2)/100;
>> VAR5 = sum((mean(Y5_hat) - Y5).^2)/100;
>> VAR6 = sum((mean(Y6_hat) - Y6).^2)/100;
>> VAR7 = sum((mean(Y7_hat) - Y7).^2)/100;
>> VAR8 = sum((mean(Y8_hat) - Y8).^2)/100;
>> VAR9 = sum((mean(Y9_hat) - Y9).^2)/100;
>> VAR10 = sum((mean(Y10_hat) - Y10).^2)/100;
VAR1 =12.5967
VAR2 =14.2544
VAR3 =11.8421
```

VAR4 =11.6762
VAR5 =8.9118
VAR6 =10.3657
VAR7 =10.6552
VAR8 =12.4832
VAR9 =13.2279
VAR10 =11.0889

True Value:
```
>> X1(:,22) = 3*mean(X1,2)-min(X1')';
```
Bias:
```
>> BAIS_X1 = sum((mean(X1_hat) - X1(:,22)).^2)/1000;

BAIS_X1 =

  120.3640

>> sum((mean(X1_hat) - X1(:,21)).^2)/1000

ans =

   12.1094
```

Dimension Deduction:

Prepare Dataset

```
data <- read.csv('communities.data', header = FALSE)
name_data <- read.delim('names', header = FALSE, sep = ' ')
names(data) <- name_data[,2]
data['communityname'] <- NULL

for(i in 1:ncol(data)){
  data[,i] <- as.numeric(data[,i])
  data[data[,i] == '?',i] <- mean(data[,i], na.rm = TRUE)
}

train_ind <- sample(seq_len(nrow(data)), size = (0.6 * nrow(data)))
train <- data[train_ind, ]
test <- data[-train_ind, ]

TS <- train
VS <- test
```

I Baseline Regression Model:

```
> ptm <- proc.time()
> fit_model <- lm(data = TS, ViolentCrimesPerPop ~ .)
> proc.time() - ptm
   user  system elapsed
  0.044   0.002   0.049

> summary(fit_model)

Call:
lm(formula = ViolentCrimesPerPop ~ ., data = TS)

Residuals:
     Min       1Q   Median       3Q      Max
-0.46219 -0.07136 -0.01125  0.05118  0.71048

Coefficients: (1 not defined because of singularities)
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.569e-01  2.657e-01   2.848  0.00448 **
state               -2.163e-04  3.362e-04  -0.643  0.52015
county              -3.965e-04  2.010e-04  -1.973  0.04879 *
community           -3.030e-05  2.635e-05  -1.150  0.25035
communityname       -2.870e-06  8.646e-06  -0.332  0.74000
fold                -1.120e-03  1.402e-03  -0.799  0.42433
population          -3.120e-01  5.432e-01  -0.574  0.56581
householdsize       -2.825e-02  1.185e-01  -0.238  0.81160
racepctblack         1.041e-01  6.778e-02   1.536  0.12478
racePctWhite        -7.515e-02  7.846e-02  -0.958  0.33841
racePctAsian        -4.913e-02  4.612e-02  -1.065  0.28696
racePctHisp         -3.139e-02  7.236e-02  -0.434  0.66456
agePct12t21          7.638e-02  1.442e-01   0.530  0.59646
agePct12t29         -2.437e-01  2.047e-01  -1.190  0.23426
agePct16t24         -9.325e-02  2.151e-01  -0.433  0.66478
agePct65up           1.516e-02  1.389e-01   0.109  0.91314
numbUrban            9.494e-02  5.367e-01   0.177  0.85964
pctUrban             2.247e-02  2.115e-02   1.062  0.28830
medIncome           -3.040e-01  2.257e-01  -1.347  0.17820
pctWWage            -4.544e-02  1.200e-01  -0.379  0.70499
pctWFarmSelf         4.839e-02  2.658e-02   1.820  0.06900 .
pctWInvInc          -1.379e-02  9.114e-02  -0.151  0.87980
pctWSocSec           1.745e-01  1.468e-01   1.189  0.23475
pctWPubAsst          4.974e-02  6.206e-02   0.801  0.42304
pctWRetire          -1.499e-01  5.047e-02  -2.970  0.00304 **
```

medFamInc	2.388e-01	2.096e-01	1.139	0.25489	
perCapInc	2.961e-01	2.561e-01	1.156	0.24774	
whitePerCap	-4.926e-01	2.050e-01	-2.402	0.01645	*
blackPerCap	2.445e-03	3.275e-02	0.075	0.94049	
indianPerCap	-2.703e-02	2.609e-02	-1.036	0.30044	
AsianPerCap	2.037e-02	2.546e-02	0.800	0.42386	
OtherPerCap	2.387e-04	2.523e-04	0.946	0.34426	
HispPerCap	1.771e-02	3.303e-02	0.536	0.59189	
NumUnderPov	3.988e-01	1.993e-01	2.000	0.04571	*
PctPopUnderPov	-4.019e-01	8.363e-02	-4.806	1.76e-06	***
PctLess9thGrade	-1.875e-01	9.049e-02	-2.072	0.03851	*
PctNotHSGrad	1.113e-01	1.288e-01	0.864	0.38778	
PctBSorMore	3.255e-02	1.018e-01	0.320	0.74928	
PctUnemployed	8.236e-03	5.447e-02	0.151	0.87985	
PctEmploy	7.897e-02	1.059e-01	0.745	0.45615	
PctEmplManu	-5.958e-02	4.337e-02	-1.374	0.16981	
PctEmplProfServ	-6.229e-02	5.413e-02	-1.151	0.25012	
PctOccupManu	7.674e-02	7.223e-02	1.062	0.28827	
PctOccupMgmtProf	1.656e-01	1.109e-01	1.493	0.13582	
MalePctDivorce	3.253e-01	3.315e-01	0.981	0.32671	
MalePctNevMarr	2.110e-01	8.641e-02	2.442	0.01476	*
FemalePctDiv	8.940e-02	4.096e-01	0.218	0.82725	
TotalPctDiv	-4.804e-01	6.912e-01	-0.695	0.48712	
PersPerFam	3.660e-02	2.253e-01	0.162	0.87098	
PctFam2Par	1.763e-01	2.071e-01	0.851	0.39498	
PctKids2Par	-5.674e-01	2.045e-01	-2.775	0.00562	**
PctYoungKids2Par	2.259e-02	6.473e-02	0.349	0.72718	
PctTeen2Par	-2.906e-02	5.680e-02	-0.512	0.60900	
PctWorkMomYoungKids	-1.200e-02	6.271e-02	-0.191	0.84828	
PctWorkMom	-5.944e-02	7.149e-02	-0.831	0.40591	
NumIlleg	-1.853e-01	1.715e-01	-1.080	0.28022	
PctIlleg	1.937e-01	6.463e-02	2.996	0.00280	**
NumImmig	-2.875e-01	1.067e-01	-2.695	0.00715	**
PctImmigRecent	1.421e-02	5.450e-02	0.261	0.79432	
PctImmigRec5	-7.550e-02	8.888e-02	-0.849	0.39582	
PctImmigRec8	7.832e-02	1.037e-01	0.755	0.45018	
PctImmigRec10	-1.475e-02	7.874e-02	-0.187	0.85140	
PctRecentImmig	-2.620e-02	1.666e-01	-0.157	0.87507	
PctRecImmig5	1.458e-01	2.978e-01	0.490	0.62456	
PctRecImmig8	-3.359e-01	3.778e-01	-0.889	0.37416	
PctRecImmig10	3.384e-01	3.026e-01	1.118	0.26363	
PctSpeakEnglOnly	-1.695e-01	8.786e-02	-1.929	0.05394	.
PctNotSpeakEnglWell	-2.443e-01	9.093e-02	-2.687	0.00733	**
PctLargHouseFam	1.064e-01	2.952e-01	0.360	0.71864	
PctLargHouseOccup	-2.950e-01	3.112e-01	-0.948	0.34343	

PersPerOccupHous	7.882e-01	3.239e-01	2.434	0.01511	*
PersPerOwnOccHous	-2.973e-01	2.168e-01	-1.371	0.17072	
PersPerRentOccHous	-2.537e-01	1.028e-01	-2.468	0.01375	*
PctPersOwnOccup	-5.593e-01	4.608e-01	-1.214	0.22509	
PctPersDenseHous	1.492e-01	1.011e-01	1.476	0.14012	
PctHousLess3BR	9.771e-02	7.957e-02	1.228	0.21970	
MedNumBR	3.413e-02	2.575e-02	1.325	0.18531	
HousVacant	1.355e-01	9.744e-02	1.390	0.16473	
PctHousOccup	-6.256e-02	4.063e-02	-1.540	0.12395	
PctHousOwnOcc	3.619e-01	4.845e-01	0.747	0.45521	
PctVacantBoarded	4.748e-02	2.859e-02	1.661	0.09702	.
PctVacMore6Mos	-7.256e-02	3.341e-02	-2.172	0.03007	*
MedYrHousBuilt	4.552e-02	3.884e-02	1.172	0.24139	
PctHousNoPhone	1.246e-01	4.818e-02	2.585	0.00986	**
PctWOFullPlumb	-1.494e-02	2.702e-02	-0.553	0.58058	
OwnOccLowQuart	-3.032e-01	2.789e-01	-1.087	0.27730	
OwnOccMedVal	1.363e-01	4.254e-01	0.320	0.74873	
OwnOccHiQuart	9.126e-02	2.249e-01	0.406	0.68495	
RentLowQ	-2.438e-01	8.668e-02	-2.813	0.00499	**
RentMedian	1.448e-01	2.053e-01	0.705	0.48070	
RentHighQ	-4.396e-02	1.112e-01	-0.395	0.69256	
MedRent	2.310e-01	1.699e-01	1.360	0.17413	
MedRentPctHousInc	6.127e-02	4.330e-02	1.415	0.15734	
MedOwnCostPctInc	-1.084e-01	4.584e-02	-2.366	0.01818	*
MedOwnCostPctIncNoMtg	-4.980e-02	3.370e-02	-1.478	0.13982	
NumInShelters	-4.182e-02	9.622e-02	-0.435	0.66396	
NumStreet	2.880e-01	6.891e-02	4.179	3.16e-05	***
PctForeignBorn	7.483e-02	1.200e-01	0.624	0.53301	
PctBornSameState	-4.970e-03	5.323e-02	-0.093	0.92563	
PctSameHouse85	-4.721e-02	7.404e-02	-0.638	0.52380	
PctSameCity85	3.937e-02	5.062e-02	0.778	0.43692	
PctSameState85	6.800e-02	5.565e-02	1.222	0.22198	
LemasSwornFT	-1.080e-02	9.708e-03	-1.113	0.26598	
LemasSwFTPerPop	3.902e-03	6.665e-03	0.585	0.55841	
LemasSwFTFieldOps	1.969e-05	3.069e-03	0.006	0.99488	
LemasSwFTFieldPerPop	3.635e-03	6.405e-03	0.568	0.57047	
LemasTotalReq	-6.471e-03	4.016e-03	-1.611	0.10746	
LemasTotReqPerPop	-1.258e-03	3.023e-03	-0.416	0.67730	
PolicReqPerOffic	4.022e-03	1.718e-03	2.341	0.01940	*
PolicPerPop	NA	NA	NA	NA	
RacialMatchCommPol	-4.102e-04	6.526e-04	-0.629	0.52979	
PctPolicWhite	-1.960e-04	1.210e-03	-0.162	0.87127	
PctPolicBlack	2.815e-03	1.753e-03	1.606	0.10861	
PctPolicHisp	2.389e-03	2.009e-03	1.189	0.23457	
PctPolicAsian	2.083e-03	1.053e-03	1.978	0.04820	*

```
PctPolicMinor        -2.870e-03  2.500e-03  -1.148  0.25135
OfficAssgnDrugUnits   1.867e-03  4.943e-03   0.378  0.70575
NumKindsDrugsSeiz    -2.069e-03  3.851e-03  -0.537  0.59114
PolicAveOTWorked      4.180e-04  5.433e-04   0.769  0.44192
LandArea             -1.807e-02  6.633e-02  -0.272  0.78532
PopDens               3.921e-02  4.018e-02   0.976  0.32931
PctUsePubTrans       -4.070e-02  3.089e-02  -1.318  0.18793
PolicCars             2.306e-03  1.600e-03   1.441  0.14982
PolicOperBudg         1.357e-02  8.707e-03   1.558  0.11950
LemasPctPolicOnPatr  -8.625e-04  9.804e-04  -0.880  0.37920
LemasGangUnitDeploy   4.970e-03  1.279e-02   0.389  0.69767
LemasPctOfficDrugUn  -1.040e-01  6.224e-02  -1.672  0.09487 .
PolicBudgPerPop      -7.280e-03  2.984e-03  -2.440  0.01485 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1306 on 1069 degrees of freedom
Multiple R-squared:  0.7318,	Adjusted R-squared:  0.7002
F-statistic: 23.15 on 126 and 1069 DF,  p-value: < 2.2e-16
```

Evaluating the linear model:

```
pred <- predict.lm(fit_model, VS[,1:126])

SSE <- sum((VS$ViolentCrimesPerPop - pred)^2)
RMSE <- sqrt(mean((VS$ViolentCrimesPerPop - pred)^2))
RSE <- sum((VS$ViolentCrimesPerPop -
pred)^2)/sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
SST <- sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
R_square <- 1-SSE/SST

> SSE
[1] 15.23605

> RMSE
[1] 0.1381767

> RSE
[1] 0.3811096

> R_square
[1] 0.6188904
```

II Feature Selection: Sequential Subset Selection

feature selection is the process of selecting a subset of relevant features for use in model construction.

```
model_step <- step(fit_model,direction = "both")
> proc.time() - ptm
   user  system elapsed
110.339   2.492 112.627
ptm <- proc.time()
model_step <- step(fit_model, direction = "backward")
proc.time() – ptm
> proc.time() - ptm
   user  system elapsed
 81.553   1.603  83.067
```

At the beginning, the algorithm chooses one variable that makes the error minimum. On every iteration, the algorithm adds a variable into the model, if the error less than before, then go to next iteration. The program will stop until there is no variable can be added in the model to make the error less than before.

```
> model_step

Call:
lm(formula = ViolentCrimesPerPop ~ county + community + population +
    racepctblack + agePct12t29 + pctUrban + pctWFarmSelf + pctWSocSec +
    pctWRetire + perCapInc + whitePerCap + NumUnderPov + PctPopUnderPov +
    PctLess9thGrade + PctNotHSGrad + PctOccupMgmtProf + MalePctDivorce +
    MalePctNevMarr + TotalPctDiv + PctKids2Par + PctWorkMom +
    NumIlleg + PctIlleg + NumImmig + PctRecImmig10 + PctSpeakEnglOnly +
    PctNotSpeakEnglWell + PctLargHouseOccup + PersPerOccupHous +
    PersPerOwnOccHous + PersPerRentOccHous + PctPersOwnOccup +
    PctPersDenseHous + PctHousLess3BR + MedNumBR + HousVacant +
    PctHousOccup + PctVacantBoarded + PctVacMore6Mos + PctHousNoPhone +
    OwnOccLowQuart + RentLowQ + MedRent + MedRentPctHousInc +
    MedOwnCostPctInc + MedOwnCostPctIncNoMtg + NumStreet + PctSameState85 +
    LemasSwFTFieldPerPop + LemasTotalReq + PolicReqPerOffic +
    PctPolicBlack + PctPolicAsian + PctUsePubTrans + PolicOperBudg +
    LemasPctPolicOnPatr + LemasPctOfficDrugUn + PolicBudgPerPop,
    data = TS)

prediction <- predict(model_step, newdata = VS[,1:126])

SSE <- sum((VS$ViolentCrimesPerPop - prediction)^2)
RMSE <- sqrt(mean((VS$ViolentCrimesPerPop - prediction)^2))
RSE <- sum((VS$ViolentCrimesPerPop -
prediction)^2)/sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
SST <- sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
R_square <- 1-SSE/SST
```

```
> SSE
[1] 15.54977
> RMSE
[1] 0.139592
> RSE
[1] 0.3889568
> R_square
[1] 0.6110432
```

III Feature Selection: Ranking Attributes

The caret R package provides tools automatically report on the relevance and importance of attributes in the data and can select the most important features out.

The Caret R package provides the findCorrelation which will analyze a correlation matrix of your data's attributes report on attributes that can be removed.

```
# load the library
library(mlbench)
library(caret)

varimp <- varImp(fit_model)
varimp[,'names'] <- rownames(varimp)
imp <- varimp[order(varimp$Overall,decreasing = TRUE),]
top50 <- head(imp,50)$names

var <- top50[1]
for (i in 2:length(top50)){
  var <- paste(var,top50[i],sep = "+")
}

var
ptm <- proc.time()
rank_model <- lm(data = TS,ViolentCrimesPerPop ~
PctPopUnderPov+NumStreet+PctIlleg+pctWRetire+RentLowQ+PctKids2Par+NumImm
ig+PctNotSpeakEnglWell+PctHousNoPhone+PersPerRentOccHous+MalePctNevMarr+
PolicBudgPerPop+PersPerOccupHous+whitePerCap+MedOwnCostPctInc+PolicReqP
erOffic+PctVacMore6Mos+PctLess9thGrade+NumUnderPov+PctPolicAsian+county+P
ctSpeakEnglOnly+pctWFarmSelf+LemasPctOfficDrugUn+PctVacantBoarded+LemasT
otalReq+PctPolicBlack+PolicOperBudg+PctHousOccup+racepctblack+PctOccupMgm
tProf+MedOwnCostPctIncNoMtg+PctPersDenseHous+PolicCars+MedRentPctHousInc
+HousVacant+PctEmplManu+PersPerOwnOccHous+MedRent+medIncome+MedNum
BR+PctUsePubTrans+PctHousLess3BR+PctSameState85+PctPersOwnOccup+agePc
t12t29+PctPolicHisp+pctWSocSec+MedYrHousBuilt+perCapInc )
```

```
proc.time() – ptm
user  system elapsed
  0.015   0.000   0.015

summary(new_model)
Residuals:
     Min       1Q   Median       3Q      Max
-0.47630 -0.07306 -0.01298  0.05491  0.72271

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          0.7343385  0.1201645   6.111 1.35e-09 ***
PctPopUnderPov      -0.3146626  0.0602208  -5.225 2.07e-07 ***
NumStreet            0.2801725  0.0587065   4.772 2.06e-06 ***
PctIlleg             0.2607021  0.0545860   4.776 2.02e-06 ***
pctWRetire          -0.1726213  0.0422249  -4.088 4.65e-05 ***
RentLowQ            -0.2149435  0.0669519  -3.210 0.001362 **
PctKids2Par         -0.3215565  0.0774366  -4.153 3.53e-05 ***
NumImmig            -0.2248940  0.0881297  -2.552 0.010844 *
PctNotSpeakEnglWell -0.1134171  0.0683330  -1.660 0.097234 .
PctHousNoPhone       0.1095215  0.0411161   2.664 0.007837 **
PersPerRentOccHous  -0.1925826  0.0788743  -2.442 0.014771 *
MalePctNevMarr       0.1687370  0.0695235   2.427 0.015376 *
PolicBudgPerPop     -0.0001511  0.0011648  -0.130 0.896810
PersPerOccupHous     0.6815691  0.1804221   3.778 0.000166 ***
whitePerCap         -0.4840277  0.1605980  -3.014 0.002636 **
MedOwnCostPctInc    -0.1272717  0.0372309  -3.418 0.000652 ***
PolicReqPerOffic     0.0026274  0.0008393   3.131 0.001789 **
PctVacMore6Mos      -0.0562606  0.0311708  -1.805 0.071351 .
PctLess9thGrade     -0.1203951  0.0472253  -2.549 0.010921 *
NumUnderPov          0.0315342  0.1072716   0.294 0.768838
PctPolicAsian        0.0009769  0.0009081   1.076 0.282268
county              -0.0005560  0.0001723  -3.228 0.001282 **
PctSpeakEnglOnly    -0.2212188  0.0578309  -3.825 0.000138 ***
pctWFarmSelf         0.0117001  0.0237256   0.493 0.622008
LemasPctOfficDrugUn -0.0898384  0.0343711  -2.614 0.009072 **
PctVacantBoarded     0.0438030  0.0265875   1.648 0.099729 .
LemasTotalReq       -0.0056126  0.0029475  -1.904 0.057134 .
PctPolicBlack        0.0014473  0.0005734   2.524 0.011738 *
PolicOperBudg        0.0014939  0.0039267   0.380 0.703687
PctHousOccup        -0.0741608  0.0358277  -2.070 0.038683 *
racepctblack         0.1644355  0.0386006   4.260 2.21e-05 ***
PctOccupMgmtProf     0.0328050  0.0566284   0.579 0.562498
MedOwnCostPctIncNoMtg -0.0645147  0.0291060  -2.217 0.026850 *
PctPersDenseHous     0.1213619  0.0792284   1.532 0.125849
```

```
PolicCars             0.0019413  0.0012899   1.505 0.132612
MedRentPctHousInc      0.0849703  0.0382574   2.221 0.026545 *
HousVacant            0.0957398  0.0794334   1.205 0.228343
PctEmplManu          -0.0008667  0.0274983  -0.032 0.974861
PersPerOwnOccHous     -0.3803554  0.1136343  -3.347 0.000843 ***
MedRent              0.2726922  0.0883764   3.086 0.002080 **
medIncome            -0.0896886  0.1350723  -0.664 0.506821
MedNumBR              0.0349731  0.0242766   1.441 0.149968
PctUsePubTrans        -0.0110622  0.0264448  -0.418 0.675795
PctHousLess3BR         0.1135483  0.0714192   1.590 0.112137
PctSameState85         0.0538289  0.0290822   1.851 0.064438 .
PctPersOwnOccup       -0.1983685  0.0603906  -3.285 0.001052 **
agePct12t29          -0.2929206  0.0852510  -3.436 0.000612 ***
PctPolicHisp         -0.0003195  0.0007791  -0.410 0.681785
pctWSocSec            0.1821452  0.0635430   2.866 0.004227 **
MedYrHousBuilt         0.0565087  0.0310955   1.817 0.069439 .
perCapInc             0.4260721  0.1901583   2.241 0.025242 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1307 on 1145 degrees of freedom
Multiple R-squared:  0.712,          Adjusted R-squared:  0.6995
F-statistic: 56.62 on 50 and 1145 DF,  p-value: < 2.2e-16

prediction <- predict(new_model, newdata = VS)

SSE <- sum((VS$ViolentCrimesPerPop - prediction)^2)
RMSE <- sqrt(mean((VS$ViolentCrimesPerPop - prediction)^2))
RSE <- sum((VS$ViolentCrimesPerPop -
prediction)^2)/sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
SST <- sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
R_square <- 1-SSE/SST

> SSE
[1] 14.2841
> RMSE
[1] 0.1337904
> RSE
[1] 0.3572977
> R_square
[1] 0.6427023
```

IV Feature Extraction: Principal Components Analysis

Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method do so by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without changing them.

Prcomp

The function prcomp() comes with the default "stats" package.

```
pca <- prcomp(TS[,1:126], retx=TRUE, center=TRUE, scale=TRUE)
summary(pca)
Importance of components:
                        PC1    PC2    PC3     PC4     PC5     PC6    PC7    PC8    PC9   PC10
PC11    PC12    PC13
Standard deviation     5.4350 4.5121 3.6918 2.81582 2.62479 2.17477 2.0900 1.8619
1.7781 1.4910 1.32720 1.27176 1.23926
Proportion of Variance 0.2326 0.1603 0.1073 0.06243 0.05425 0.03724 0.0344 0.0273
0.0249 0.0175 0.01387 0.01274 0.01209
Cumulative Proportion  0.2326 0.3929 0.5002 0.56265 0.61689 0.65414 0.6885 0.7158
0.7407 0.7582 0.77210 0.78483 0.79693
                       PC14    PC15    PC16   PC17    PC18    PC19    PC20    PC21    PC22
PC23    PC24    PC25
Standard deviation     1.2034 1.12070 1.06611 1.0266 1.00762 0.99102 0.94475
0.93679 0.89851 0.87210 0.82693 0.81555
Proportion of Variance 0.0114 0.00989 0.00895 0.0083 0.00799 0.00773 0.00703
0.00691 0.00636 0.00599 0.00538 0.00524
Cumulative Proportion  0.8083 0.81822 0.82717 0.8355 0.84346 0.85119 0.85822
0.86513 0.87149 0.87748 0.88286 0.88810
                        PC26    PC27    PC28    PC29   PC30    PC31    PC32    PC33    PC34
PC35    PC36   PC37
Standard deviation     0.79453 0.78804 0.75758 0.73739 0.7216 0.71364 0.70110
0.69308 0.67320 0.66893 0.65279 0.6172
Proportion of Variance 0.00497 0.00489 0.00452 0.00428 0.0041 0.00401 0.00387
0.00378 0.00357 0.00352 0.00336 0.0030
Cumulative Proportion  0.89307 0.89796 0.90248 0.90676 0.9109 0.91487 0.91874
0.92252 0.92609 0.92962 0.93297 0.9360
```

There are 128 components were constructed. The minimum number of components needed to capture at least 90% of the data variance is 28, because the cumulative proportion of PC28 is 0.90248.

Prepare the pca data:

```
newdata <- pca$x[,1:28]
newdata <- data.frame(newdata)
newdata[,29] <- TS[,127]
```

Fit a linear model with pca data:

```
fitmodel <- lm(data = newdata, V29 ~ .)
user  system elapsed
  0.103   0.008   0.114
```

Transform the validation data using pca model:

```
pred.vs <- predict(pca, VS[,1:126])
pred.vs <- data.frame(pred.vs)
pred.vs <- pred.vs[,1:28]
```

Validation the result:

```
prediction <- predict(fitmodel, newdata = pred.vs)
SSE <- sum((VS$ViolentCrimesPerPop - prediction)^2)
RMSE <- sqrt(mean((VS$ViolentCrimesPerPop - prediction)^2))
RSE <- sum((VS$ViolentCrimesPerPop -
prediction)^2)/sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
SST <- sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
R_square <- 1-SSE/SST

> SSE
[1] 14.70018
> RMSE
[1] 0.135725
> RSE
[1] 0.3677056
> R_square
[1] 0.6322944
```

V Feature Extraction: Factor Analysis (FA)

```
library(psych)
ptm <- proc.time()
fa_fit <- factor.pa(TS[,1:126], nfactors=30)
fa_data <- predict(object = fa_fit, data = TS[,1:126])
# set 'ViolentCrimesPerPop' column
fa_data <- data.frame(fa_data)
fa_data[,31] <- TS[,127]
colnames(fa_data)[31] <- "ViolentCrimesPerPop"
fit_fa_model <- lm(data = fa_data, ViolentCrimesPerPop ~ .)
proc.time() - ptm

user  system elapsed
  0.593   0.020   0.613

# transfer Validation data using FA model
fa_VS_data <- predict(object = fa_fit, data = VS[,1:126])
fa_VS_data <- data.frame(fa_VS_data)
```

```
fa_VS_data[,31] <- VS[,127]
colnames(fa_VS_data)[31] <- "ViolentCrimesPerPop"
prediction <- predict(fit_fa_model, newdata = fa_VS_data[,1:30])

SSE <- sum((VS$ViolentCrimesPerPop - prediction)^2)
RMSE <- sqrt(mean((VS$ViolentCrimesPerPop - prediction)^2))
RSE <- sum((VS$ViolentCrimesPerPop -
prediction)^2)/sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
SST <- sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
R_square <- 1-SSE/SST

> SSE
[1] 14.83396
> RMSE
[1] 0.1363412
> RSE
[1] 0.3710519
> R_square
[1] 0.6289481
```

VI Comparison of Results

	Baseline	Sequential Subset Selection	Relief	PCA	FA
Number of attributes used to construct the linear regression model	127	127	127	28	Has Error
Number of attributes appearing in the linear regression model	127	58	50	28	Has Error
Time taken constructing the linear regression model	0.063	0.265	0.043	0.021	Has Error
Sum of Square Errors(SSE)	15.79304	15.66877	16.00039	15.22379	Has Error
Root Mean Square Error(RMSE)	0.1406797	0.1401251	0.1416002	0.1381211	Has Error

Relative Square Error(RSE)	0.3930557	0.3899629	0.3982162	0.3788883	Has Error
Coeffient of Determination(R^2)	0.6069443	0.6100371	0.6017838	0.6211117	Has Error

From the table we can see that using PCA can have the highest R-Square and lowest RMSE. Using Subset Selection is the next but it used 58 features to do the linear regression model. Using top 50 features to do linear regression model is similar to baseline model.