```
#### Lab 1 - Assignment 3 - Feature selection by cross-validation in a
linear model

### Functions
# Returns linear regression model of given X and Y
linear_model=function(..X, ..Y){
  X = cbind(1, ..X)
  β = solve(t(X)%*%X)%*%t(X)%*%..Y
  return(β)
}

CV=function(..X, ..Y, K){
  # Setup
  n = length(..Y)
  p = ncol(..X)
  set.seed(12345)
  ids = sample(n,n)
  X = ..X[ids,]
  Y = ..Y[ids]
  width = floor(n/K)
  MSEs = numeric(2^p-1)
  N.features = numeric(2^p-1)
  features = list()
  current = 0

  # Assuming 5 features:

  for (f1 in 0:1){
    for (f2 in 0:1){
      for(f3 in 0:1){
        for(f4 in 0:1){
          for(f5 in 0:1){
            model= c(f1,f2,f3,f4,f5)
            if (sum(model)==0) next()
            SSE=0

            for (k in 1:K){
              # Select indices
              indices = 1:width + (k-1)*width
              if(k==K) indices = ((k-1)*width+1):n

              # Train model
              X.training = X[-indices, which(model==1)]
              Y.training = Y[-indices]
              β.k = linear_model(X.training, Y.training)

              # Make predictions
              X.k = cbind(1, X[indices, which(model==1)])
              Y.k = Y[indices]
              Yfit.k = X.k %*% β.k

              # Calculate Error
              SSE=SSE+sum((Yfit.k-Y.k)^2)
            }
            # Store performance of current feature selection
            current = current+1
            MSEs[current] = SSE/n
            N.features[current] = sum(model)
            features[[current]] = model
          }
        }
      }
    }
  }

  # Plot MSE against number of features
  plot(N.features, MSEs)

  # Return info about best feature selection
  i=which.min(MSEs)
  return(list(CV=MSEs[i], Features=features[[i]],
Feature_Names=colnames(X)[which(features[[i]]==1)]))
}

### Implementation
X.swiss = as.matrix(swiss[,2:6])
Y.swiss = swiss[[1]]
N.folds = 5
CV(X.swiss, Y.swiss, N.folds)
```