In [1]:
```
using Gurobi, CSV, DataFrames, JuMP, LinearAlgebra, Distributions, GLMNet, Rando
```

In [2]:
```
Xtrain = CSV.read("airfoil_X_train.csv", DataFrame, header=1);
ytrain = CSV.read("airfoil_Y_train.csv", DataFrame, header=1)[:,1];
Xtest = CSV.read("airfoil_X_test.csv", DataFrame, header=1);
ytest = CSV.read("airfoil_Y_test.csv", DataFrame, header=1)[:,1];
```

3.1 Question 1: Holistic Regression

3.1a)

In [3]:
```
cv = glmnetcv(Matrix(Xtrain), ytrain)
GLMNet.coef(cv)
```

Out[3]:
```
15-element Vector{Float64}:
 -0.12135865302357253
 -0.20171078656217045
 -0.34545429509859027
  0.06345766062718952
 -0.002724358478913054
 -0.0414582918628037
 -0.45546592619148085
  0.08180247107368721
 -0.4056202426222409
  0.13059609061333535
 -0.13816204192771966
 -0.1435956235942216
  0.18090319386557993
 -0.2966023001087412
  0.3642489167048034
```

In [4]:
```
function compute_mse(X, y, beta)
    n,p = size(X)
    return sum((y .- X*beta[1:p]).^2)/n
end

function compute_mse_wb0(X, y, beta, beta_0)
    n,p = size(X)
    return sum((y .- X*beta .- beta_0).^2)/n
end
```

Out[4]:
```
compute_mse_wb0 (generic function with 1 method)
```

In [5]:
```
function rsq(X, y, beta)
    n,p = size(X)
    return 1-sum((y .- X*beta[1:p]).^2)/sum((y .- (y/n)).^2)
end
```

Out[5]:
```
rsq (generic function with 1 method)
```

In [6]:
```
compute_mse(Matrix(Xtest), ytest, GLMNet.coef(cv))
```

Out[6]:
```
0.3749824414598864
```

In [7]:
```
rsq(Matrix(Xtest), ytest, GLMNet.coef(cv))
```

Out[7]:
```
0.635629817698532
```

3.1b)

In [8]:
```
comat = cor(Matrix(Xtrain))
```

Out[8]:
```
15×15 Matrix{Float64}:
  1.0         -0.278659   -0.0301151  …   0.0367723  -0.204695   -0.221726
 -0.278659     1.0        -0.500638      -0.426344    0.371662    0.729439
 -0.0301151  -0.500638     1.0            0.881465    0.203822   -0.216972
  0.126902     0.0426135   0.0242601      0.409468   -0.0096227   0.244131
 -0.245811     0.762491   -0.227926      -0.207703    0.817769    0.926974
  0.376657     0.487474   -0.286753   …  -0.21805     0.116095    0.308391
  0.680399    -0.34138     0.467137       0.492894   -0.0313951  -0.197011
  0.944555    -0.240804   -0.0059982      0.137558   -0.184829   -0.163565
  0.250875     0.443983   -0.119055      -0.0911413   0.468395    0.528344
 -0.26623      0.467794    0.2395         0.240433    0.809066    0.61753
 -0.229541     0.909715   -0.443013   …  -0.293285    0.329069    0.786103
 -0.221854     0.832278   -0.340088      -0.297481    0.66444     0.901821
  0.0367723   -0.426344    0.881465       1.0         0.164615   -0.12064
 -0.204695     0.371662    0.203822       0.164615    1.0         0.755483
 -0.221726     0.729439   -0.216972      -0.12064     0.755483    1.0
```

In [9]:
```
function cor_var_mat(X, rho)
    mat = []
    c = cor(X)
    n = size(c)[1]

    for i in 1:n
        vec = c[i,:]
        for j in 1:n
            if i != j
                if vec[j] >= rho
                    push!(mat,(i,j,vec[j]))
                end
            end
        end
    end
    return mat
end
#Kyle Maulden showed me the helper function to generalize the one I built myself
```

Out[9]:
```
cor_var_mat (generic function with 1 method)
```

In [10]:
```
for i in 1:15
    for j in 1:15
        if i>=j
            if i!=j
                if abs(comat[i,j]) > .7
                    print("(")
```

```
                        print(i)
                        print(",")
                        print(j)
                        print("),")
                    end
                end
            end
        end
    end
```

```
(5,2),(8,1),(9,6),(11,2),(12,2),(12,5),(12,11),(13,3),(14,5),(14,10),(15,2),(15,
5),(15,11),(15,12),(15,14),
```

delete 15

delete 14

delete 12

delete 13

delete 11

delete 9

delete 8

delete 5

In [11]:

```
Xtrain_s = select(Xtrain, Not([:angle_X_velocity, :angle_X_displacement, :length
Xtest_s = select(Xtest, Not([:angle_X_velocity, :angle_X_displacement, :length_X
```

In [12]:

```
cv2 = glmnetcv(Matrix(Xtrain_s), ytrain)
GLMNet.coef(cv2)
```

Out[12]:

```
7-element Vector{Float64}:
 -0.026516132103133614
 -0.2465980601032888
 -0.1876660300526013
  0.22694875062206513
 -0.3155679770195879
 -0.5171711262852544
 -0.07694596169357004
```

In [13]:

```
print("R-squared:\n")
print(rsq(Matrix(Xtest_s), ytest, GLMNet.coef(cv2)))
print("\nMSE:\n")
print(compute_mse(Matrix(Xtest_s), ytest, GLMNet.coef(cv2)))
```

```
R-squared:
0.5760754068072742
MSE:
0.43627137090700685
```

3.1c)

In [37]:
```
function trans_x(X;eps = 1e-10)
    n,p = size(X)
    X_t = zeros((n, 4*p))
    for j in 1:p
        X_t[:,4(j-1) + 1] = X[:,j]
        X_t[:,4(j-1) + 2] = X[:,j].^2
        X_t[:,4(j-1) + 3] = sqrt.(abs.(X[:,j]))
        X_t[:,4(j-1) + 4] = log.(abs.(X[:,j]) .+ eps)
    end
    return X_t
end
```

Out[37]: trans_x (generic function with 1 method)

In [38]:
```
function holy_reg(X,y,lambda,rho,M,k;solver_output=0)

    n,p = size(X)
    p_bar = p / 4
    hc = cor_var_mat(X, rho)

    # Build model
    model = Model(Gurobi.Optimizer)
    set_optimizer_attribute(model, "OutputFlag", solver_output)
    set_optimizer_attribute(model, "IntFeasTol", 1e-9)

    # Insert variables
    @variable(model,beta_i[i=1:p])
    @variable(model,beta_0)
    @variable(model,beta_i_reg[i=1:p])
    @variable(model,z_i[i=1:p],Bin)

    # L1 Constraint
    @constraint(model,[i=1:p], beta_i_reg[i] >= beta_i[i])
    @constraint(model,[i=1:p], beta_i_reg[i] >= -beta_i[i])

    # Sparsity Constraints
    # Equation 3.2
    @constraint(model,[i=1:p], -M*z_i[i] <= beta_i[i])
    @constraint(model,[i=1:p], beta_i[i] <= M*z_i[i])
    # Equation 3.3
    @constraint(model,sum(z_i) <= k)

    # Equation 3.4 - Limiting Model to one transformation of an x variable
    for j in 1:p_bar
        @constraint(model,z_i[Int(4(j-1)+1)] + z_i[Int(4(j-1)+2)] + z_i[Int(4(j-
    end

    # Equation 3.5 - Limit Pairwise Collinearity
    for pairwise in hc
        @constraint(model,z_i[Int(pairwise[1])] + z_i[Int(pairwise[2])] <= 1)
    end

    #Objective
    @objective(model,Min, (1/2)*sum((y .- X*beta_i .- beta_0).^2) + lambda*sum(b

    # Optimize
    optimize!(model)

    # Return estimated betas
```

```
    return (value.(beta_i),value.(beta_0))

end
```

Out[38]: holy_reg (generic function with 1 method)

In [39]:

```
Xtrain_t = trans_x(Xtrain)
beta_op, beta_0_op = holy_reg(Xtrain_t,ytrain,.1,.7,999,10)
print(beta_op)
```

```
Academic license - for non-commercial use only - expires 2022-08-19
[0.0, 0.0, 0.0, -0.002354903002875301З, 0.0, 0.0, 0.0, -0.003727925801007079, 0.
0, 0.0, 0.0, -0.007326579873714567 4, 0.0, 0.0, 0.0, 0.06623439122152379, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.002425337806268929, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, -0.006049344617990648, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, -0.008791535383299846, 0.0, 0.0, 0.0, 0.006172074478869451 5, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, -0.005203923857426994, 0.0, 0.0, 0.0, -0.003726921913562803
8, 0.0, 0.0, 0.0, 0.0]
```

In [40]:

```
function holy_reg_cv(X,y,lambdas,rho,M,k,folds;solver_output=0)
    n,p = size(X)
    cut = convert(Int,floor(n/folds)) #floor takes the integer part
    lam_i_error = zeros(length(lambdas))

    #cross validating each each lambda value
    for (i,lambda) in enumerate(lambdas)
        Random.seed!(i)
        errors = zeros(folds)
        for (j,fold) in enumerate(collect(Kfold(n,folds)))
            Xtrain, ytrain = X[fold,:], y[fold]
            val_ind = [x for x in 1:n if !(x in fold)]
            Xvalid, yvalid = X[val_ind,:], y[val_ind]
            beta, beta_0 = holy_reg(X,y,lambda,rho,M,k;solver_output=0)
            mse = compute_mse_wb0(Matrix(Xvalid), yvalid, beta, beta_0)
            errors[j] = mse
        end
        avg_error = mean(errors)
        lam_i_error[i] = avg_error
    end

    #reporting the best performing lambda
    op_lam = argmin(lam_i_error)
    beta_op, beta_0_op = holy_reg(X,y,lambdas[op_lam],rho,M,k)
    println("Best Lambda:\n")
    print(lambdas[op_lam])
    println("\nOptimal Betas")
    print(beta_op)
    return beta_op, beta_0_op, lambdas[op_lam], lam_i_error
end
```

Out[40]: holy_reg_cv (generic function with 1 method)

In [44]:

```
lam = [0.01,0.5,1,1.5,2,2.5,3,3.5, 4, 4.5, 5,10]
op_reg = holy_reg_cv(Xtrain_t,ytrain,lam,.7,999,10,10)
```

```
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
```

```
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
```

```
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Academic license - for non-commercial use only - expires 2022-08-19
Best Lambda:

3.5
Optimal Betas
[0.0, 0.0, 0.0, -0.00224525908709585З, 0.0, 0.0, 0.0, -0.003595843702477886, 0.
0, 0.0, 0.0, -0.007288415927921829, 0.0, 0.0, 0.0, 0.0661456462913280 9, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.002198090028484860 5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.005899227308464097, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, -0.008738308219167589, 0.0, 0.0, 0.0, 0.006049171927637974, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, -0.004858533836158934, 0.0, 0.0, 0.0, -0.003557995748794141
7, 0.0, 0.0, 0.0, 0.0]
([0.0, 0.0, 0.0, -0.002245259087095853, 0.0, 0.0, 0.0, -0.003595843702477886, 0.
```

Out[44]:
```
0, 0.0  …  0.0, -0.004858533836158934, 0.0, 0.0, 0.0, -0.0035579957487941417, 0.
0, 0.0, 0.0, 0.0], 0.3230178173179774, 3.5, [0.11399959180718311, 0.113068694479
2469, 0.11354830533133568, 0.11350982911377963, 0.11474278241441294, 0.113908505
55743341, 0.11394129412917697, 0.11300958775838284, 0.11303244317585763, 0.11513
324083850489, 0.1139857176559774, 0.11370508692004115])
```

In [45]:
```
beta_hat, beta_0_hat = holy_reg(Xtrain_t,ytrain,3.5,.7,999,10)
Xtest_t = trans_x(Xtest)
println("Holistic Regression MSE:")
compute_mse_wb0(Xtest_t,ytest,beta_hat,beta_0_hat)
```

```
Academic license - for non-commercial use only - expires 2022-08-19
Holistic Regression MSE:
```

Out[45]:
```
0.15824889654214308
```

3.1d)

Comparing with Ananya, we had varying models. Her optimal λ value was 15 while mine was 3.5. This is most likely a result of her doing 1-fold CV as opposed to my 10-fold cross validation. I'd expect our values would be more similar if increased her k for CV. As a result, our models coefficients varied in the subset and values.

Conversely, Iggy Siegel had an optimal lambda value of 5. The subset our regressions selected were very similar but varied slightly. Again, this was likely due to his 5-fold cross validation.

3.2 Question 2: Robust Classification

In [46]:
```
Xtrain = Matrix(CSV.read("votes_X_train.csv",DataFrame; header=true));
Xtest = Matrix(CSV.read("votes_X_test.csv",DataFrame; header=true));
ytrain = Vector(CSV.read("votes_Y_train.csv",DataFrame; header=true)[:,1]);
ytest = Vector(CSV.read("votes_Y_test.csv",DataFrame; header=true)[:,1]);
```

3.2a)

In [47]:
```
function robust_class_3a(X,y,C)
    n,p = size(X)

    model = Model(Gurobi.Optimizer)
    set_optimizer_attribute(model, "OutputFlag", 0)

    # Insert variables
    @variable(model,w[i=1:p])
    @variable(model,w_reg[i=1:p])
    @variable(model,z[i=1:n])

    #epigraph of maximization function
    @constraint(model,[i=1:n], z[i] >= 0)
    @constraint(model,[i=1:n], z[i] >= 1-y[i]*dot(X[i,:],w))

    #L1 Constraint
    @constraint(model,[i=1:p], w_reg[i] >= w[i])
    @constraint(model,[i=1:p], w_reg[i] >= -w[i])
```

```
    c_ypos = n / count(item -> (item==1), y)
    c_yneg = n / count(item -> (item==-1), y)

    @objective(model, Min, C * sum((y[i]==1 ? c_ypos : c_yneg)*z[i] for i=1:n) +

    optimize!(model)
    return value.(w)
end
```

Out[47]: robust_class_3a (generic function with 1 method)

In [48]:

```
function predict(X,w)
    yhat = X*w
    y_class = @.ifelse(yhat >= 0, 1, -1)
    return y_class
end
```

Out[48]: predict (generic function with 1 method)

In [49]:

```
#I got these helper function from Kyle Maulden
function class_acc(preds,y)
    n = size(y)[1]
    count = 0
    for i in 1:n
        if preds[i] == y[i]
            count = count + 1
        end
    end
    return count / n
end
```

Out[49]: class_acc (generic function with 1 method)

In [50]:

```
w = robust_class_3a(Xtrain, ytrain, 1);
preds = predict(Xtrain,w)
println("\nIn-Sample Accuracy:")
print(class_acc(preds,ytrain))
preds = predict(Xtest,w)
println("\nOut-of-Sample Accuracy:")
print(class_acc(preds,ytest))
```

```
Academic license - for non-commercial use only - expires 2022-08-19

In-Sample Accuracy:
0.9724770642201835
Out-of-Sample Accuracy:
0.9447004608294931
```

3.2b)

The uncertainty set $U = \{\Delta y \in \{0,1\}^n : e^T y \leq \Gamma\}$ represents possibility of, at most Γ, data points being mislabeled.

3.2f)

In [51]:
```
function robust_class_3f(X,y,C,Gamma,M)
    n,p = size(X)

    model = Model(Gurobi.Optimizer)
    set_optimizer_attribute(model, "OutputFlag", 0)

    # Insert variables
    @variable(model,q)
    @variable(model,r[i=1:n])
    @variable(model,phi[i=1:n])
    @variable(model,xi[i=1:n])
    @variable(model,w[i=1:p])
    @variable(model,w_reg[i=1:p])
    @variable(model,t[i=1:n],Bin)
    @variable(model,s[i=1:n],Bin)

    c_yneg = n / count(item -> (item==-1), y)
    c_ypos = n / count(item -> (item==1), y)

    #Constraints
    @constraint(model,[i=1:n], q + r[i] >= (y[i]==1 ? c_ypos : c_yneg) * (phi[i]
    @constraint(model,[i=1:n], phi[i] >= 1 + y[i]*dot(X[i,:],w))
    @constraint(model,[i=1:n], phi[i] <= 1 + y[i]*dot(X[i,:],w) + M*(1-t[i]))
    @constraint(model,[i=1:n], phi[i] >= 0)
    @constraint(model,[i=1:n], phi[i] <= M*t[i])
    @constraint(model,[i=1:n], xi[i] >= 1 - y[i]*dot(X[i,:],w))
    @constraint(model,[i=1:n], xi[i] <= 1 - y[i]*dot(X[i,:],w) + M*(1-s[i]))
    @constraint(model,[i=1:n], xi[i] >= 0)
    @constraint(model,[i=1:n], xi[i] <= M*s[i])
    @constraint(model,[i=1:p], w_reg[i] >= w[i])
    @constraint(model,[i=1:p], w_reg[i] >= -w[i])
    @constraint(model,q>=0)
    @constraint(model,[i=1:n],r[i]>=0)

    @objective(model, Min, sum(w_reg[i] for i=1:p) + C*Gamma*q + C*sum(r[i] + (y

    optimize!(model)
    return value.(w)
end
```

Out[51]:
```
robust_class_3f (generic function with 1 method)
```

In [52]:
```
w = robust_class_3f(Xtrain, ytrain, 1, 10, 999);
preds = predict(Xtrain,w)
println("\nIn-Sample Accuracy:")
print(class_acc(preds,ytrain))
preds = predict(Xtest,w)
println("\nOut-of-Sample Accuracy:")
print(class_acc(preds,ytest))
```

```
Academic license - for non-commercial use only - expires 2022-08-19

In-Sample Accuracy:
0.9678899082568807
Out-of-Sample Accuracy:
0.9493087557603687
```

3.2g)

In [53]:

```
function robust_class_3g(X,y,C,Gamma,M)
    n,p = size(X)

    model = Model(Gurobi.Optimizer)
    set_optimizer_attribute(model, "OutputFlag", 0)

    # Insert variables
    @variable(model,q)
    @variable(model,r[i=1:n])
    @variable(model,phi[i=1:n])
    @variable(model,xi[i=1:n])
    @variable(model,w[i=1:p])
    @variable(model,w_reg[i=1:p])
    @variable(model,t[i=1:n])
    @variable(model,s[i=1:n])

    c_yneg = n / count(item -> (item==-1), y)
    c_ypos = n / count(item -> (item==1), y)

    #Constraints
    @constraint(model,[i=1:n], q + r[i] >= (y[i]==1 ? c_ypos : c_yneg) * (phi[i]
    @constraint(model,[i=1:n], phi[i] >= 1 + y[i]*dot(X[i,:],w))
    @constraint(model,[i=1:n], phi[i] <= 1 + y[i]*dot(X[i,:],w) + M*(1-t[i]))
    @constraint(model,[i=1:n], phi[i] >= 0)
    @constraint(model,[i=1:n], phi[i] <= M*t[i])
    @constraint(model,[i=1:n], xi[i] >= 1 - y[i]*dot(X[i,:],w))
    @constraint(model,[i=1:n], xi[i] <= 1 - y[i]*dot(X[i,:],w) + M*(1-s[i]))
    @constraint(model,[i=1:n], xi[i] >= 0)
    @constraint(model,[i=1:n], xi[i] <= M*s[i])
    @constraint(model,[i=1:p], w_reg[i] >= w[i])
    @constraint(model,[i=1:p], w_reg[i] >= -w[i])
    @constraint(model,q>=0)
    @constraint(model,[i=1:n],r[i]>=0)

    #New constraints
    @constraint(model,[i=1:n],0<=t[i]<=1)
    @constraint(model,[i=1:n],0<=s[i]<=1)

    @objective(model, Min, sum(w_reg[i] for i=1:p) + C*Gamma*q + C*sum(r[i] + (y

    optimize!(model)
    return value.(w)
end
```

Out[53]: robust_class_3g (generic function with 1 method)

In [54]:

```
w = robust_class_3g(Xtrain, ytrain, 1, 10, 999);
preds = predict(Xtrain,w)
println("\nIn-Sample Accuracy:")
print(class_acc(preds,ytrain))
preds = predict(Xtest,w)
println("\nOut-of-Sample Accuracy:")
print(class_acc(preds,ytest))
```

```
Academic license - for non-commercial use only - expires 2022-08-19

In-Sample Accuracy:
0.9678899082568807
Out-of-Sample Accuracy:
0.9493087557603687
```

In [ ]:

```

```