%Purpose: Optimization model for the project

%This model will assume no/little interaction between the features...

%Fix one of them, keep incrementing until its optimum is reached, and then

%move on the next feature to do the same

%This is going to be done in parallel loops (additive)

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%% DECLARE PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Random Forest Parameters

numTrees = 250;

%CSV File Path Parameters

LowBudgetCSV = 'MovieData\_LowBudget.csv';

MediumBudgetCSV = 'MovieData\_MediumBudget.csv';

HighBudgetCSV = 'MovieData\_HighBudget.csv';

AllCSV = 'MovieData\_All.csv';

movieConstructionCSV = 'MovieConstruction.csv';

%Objective Function Parameters

%Store this as a vector to make it easier to pass to functions

%Framework: obj\_function\_params = [weightBoxOffice, weightMetacritic,

%meanBoxOffice, meanMetacritic, sigmaBoxOffice, sigmaMetacritic]

objFunctionParams = [0.05, 0.95, 74.99, 54.25, 91.37, 17.46];

%Optimization Model Constraint Parameters:

%Budget

budget\_UB = 100; %USER CHOSEN

budget\_LB = 0;

budget\_Step = 2;

%Theatres (widest release)

theatres\_UB = 1000; %USER CHOSEN

theatres\_LB = 0;

theatres\_Step = 5;

%Theatres (opening release)

opening\_UB = 1000; %USER CHOSEN

opening\_LB = 0;

opening\_Step = 5;

%Franchise (1 for must be franchise; 0 for must NOT be; -1 for anything)

franchise = 1; %USER CHOSEN

%Google trends

trends\_UB = 50; %USER CHOSEN

trends\_LB = 0;

trends\_Step = 2;

%Sentiment

sentiment\_UB = 1;

sentiment\_LB = -1;

sentiment\_Step = 0.1;

%Number of Best Director nominations from director:

directorNom\_LB = 0;

directorNom\_UB = 4;

directorNom\_Step = 1;

%Number of Best Director wins from director:

directorWin\_LB = 0;

directorWin\_UB = 2;

directorWin\_Step = 1;

%Number of nominated actors

numNominated\_UB = 5; %USER CHOSEN

numNominated\_LB = 0;

numNominated\_Step = 1;

%Number of winning actors

numWinning\_UB = 5; %USER CHOSEN

numWinning\_LB = 0;

numWinning\_Step = 1;

%Total number of nominations across cast

totalNominated\_UB = 20;

totalNominated\_LB = 0;

totalNominated\_Step = 2;

%Total number of winning

totalWinning\_UB = 10;

totalWinning\_LB = 0;

totalWinning\_Step = 2;

%Months (USER CHOSEN) --> 1 if allowed, 0 else

jan = 1;

feb = 1;

mar = 1;

apr = 1;

may = 1;

jun = 1;

jul = 1;

aug = 1;

sep = 1;

oct = 1;

nov = 1;

dec = 1;

%Runtime

runtime\_UB = 200; %USER CHOSEN

runtime\_LB = 0; %USER CHOSEN

%Genres (USER CHOSEN) --> 1 if allowed, 0 else

%^^^Ignore that line above for testing though...(assume all allowed)

adventure = 1;

action = 1;

animation = 1;

family = 1;

scifi = 1;

comedy = 1;

drama = 1;

romance = 1;

horror = 1;

mystery = 1;

documentary = 1;

music = 1;

thriller = 1;

crime = 1;

war = 1;

%Total Number of Languages

languages\_UB = 10; %USER CHOSEN

languages\_LB = 1;

languages\_Step = 1;

%Content Rating (USER CHOSEN) --> 1 if allowed, 0 else

PG13 = 1;

PG = 1;

R = 1;

G = 1;

NC17 = 1;

NR = 1;

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%% READ TABLES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Read all of the tables

%Framework for the multiple BOX OFFICE MODELS:

%Read in low budget data set

movieDataLow = readtable(LowBudgetCSV);

%Block out title and targets

movieDataLow.title=[];

%Read in medium budget data set

movieDataMedium = readtable(MediumBudgetCSV);

%Block out the titles and targets

movieDataMedium.title=[];

%FIX SUN

movieDataMedium.Sun = str2double(movieDataMedium.Sun);

%Read in high budget data set

movieDataHigh = readtable(HighBudgetCSV);

%Block out the titles and targets:

movieDataHigh.title=[];

%Framwork for the metacritic model(s):

%Read in data set containing ALL films

movieDataAll = readtable(AllCSV);

titles=movieDataAll.title;

movieDataAll.title=[];

%FIX SUN

movieDataAll.Sun = str2double(movieDataAll.Sun);

%Framework for the movie construction vector:

movieConstruction = readtable(movieConstructionCSV);

%movieConstruction.AdjustedDomestic = [];

%movieConstruction.metacritics=[];

%movieConstruction.title=[];

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%% TRAIN MODELS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Note: No need to split into test and train because we're using all data

%Train a model with all of the data:

%TRAIN LOW BUDGET MODEL

modelLow = TreeBagger(numTrees, movieDataLow, 'AdjustedDomestic','Method','regression');

%TRAIN MEDIUM BUDGET MODEL

modelMedium = TreeBagger(numTrees, movieDataMedium, 'AdjustedDomestic','Method','regression');

%TRAIN HIGH BUDGET MODEL

modelHigh = TreeBagger(numTrees, movieDataHigh, 'AdjustedDomestic','Method','regression');

%TRAIN METACRITIC MODEL (using all data)

modelCritic = TreeBagger(numTrees, movieDataAll, 'metacritics','Method','regression');

%%

s = rng;

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%% OPTIMIZATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Pick starting vector as the LB of all variables

movieConstruction(1, 'AdjustedBudget') = {randi([1 150])};

movieConstruction(1, 'Num\_Theatres\_Widest') = {randi([10 4000])};

% if franchise == -1

% movieConstruction(1, 'Franchise\_Raw') = {0};

% else

% movieConstruction(1, 'Franchise\_Raw') = {franchise};

% end

movieConstruction(1, 'Franchise\_Raw') = {0};

movieConstruction(1, 'Google\_Trends') = {randi([0 100])};

movieConstruction(1, 'sentiment') = {-1 + rand(1,1)\*2};

movieConstruction(1, 'director\_nom') = {randi([0 4])};

movieConstruction(1, 'director\_win') = {randi([0 2])};

movieConstruction(1, 'numNominatedActors') = {randi([0 7])};

movieConstruction(1, 'numWinningActors') = {randi([0 4])};

movieConstruction(1, 'totNumNom') = {randi([0 16])};

movieConstruction(1, 'totNumWins') = {randi([0 5])};

movieConstruction(1, 'Mon') = {1};

movieConstruction(1, 'Tue') = {0};

movieConstruction(1, 'Wed') = {0};

movieConstruction(1, 'Thurs') = {0};

movieConstruction(1, 'Fri') = {0};

movieConstruction(1, 'Sat') = {0};

movieConstruction(1, 'Sun') = {0};

movieConstruction(1, 'Jan') = {1};

movieConstruction(1, 'Feb') = {0};

movieConstruction(1, 'Mar') = {0};

movieConstruction(1, 'Apr') = {0};

movieConstruction(1, 'May') = {0};

movieConstruction(1, 'Jun') = {0};

movieConstruction(1, 'Jul') = {0};

movieConstruction(1, 'Aug') = {0};

movieConstruction(1, 'Sep') = {0};

movieConstruction(1, 'Oct') = {0};

movieConstruction(1, 'Nov') = {0};

movieConstruction(1, 'Dec') = {0};

movieConstruction(1, 'runtime') = {randi([60 300])};

movieConstruction(1, 'Adventure') = {0};

movieConstruction(1, 'Action') = {0};

movieConstruction(1, 'Animation') = {0};

movieConstruction(1, 'Family') = {0};

movieConstruction(1, 'ScienceFiction') = {0};

movieConstruction(1, 'Comedy') = {0};

movieConstruction(1, 'Drama') = {1};

movieConstruction(1, 'Romance') = {0};

movieConstruction(1, 'Horror') = {0};

movieConstruction(1, 'Mystery') = {0};

movieConstruction(1, 'Documentary') = {0};

movieConstruction(1, 'Music') = {0};

movieConstruction(1, 'Thriller') = {0};

movieConstruction(1, 'Crime') = {0};

movieConstruction(1, 'War') = {0};

movieConstruction(1, 'total\_language') = {randi([1 7])};

movieConstruction(1, 'PG\_13') = {0};

movieConstruction(1, 'PG') = {0};

movieConstruction(1, 'R') = {1};

movieConstruction(1, 'G') = {0};

movieConstruction(1, 'NC\_17') = {0};

movieConstruction(1, 'NR') = {0};

%For ease of editing, we're going to put all of the features into a cell

%array. The features will be optimized in the order that they appear in

%the vector. Thus, if the order changes or is optimized, we just need to

%change the order of entries in this vector:

feature\_order = {'Num\_Theatres\_Widest', 'AdjustedBudget', 'sentiment',...

'runtime', 'Google\_Trends', 'numNominatedActors', 'numWinningActors', 'totNumNom',...

'totNumWins', 'director\_nom', 'director\_win',...

'Franchise\_Raw', 'total\_language'};

%Get length of this array of features:

num\_features = length(feature\_order);

%%

fprintf('ghostostostostostostostost')

%Sensitivity analysis for budget (on critical score):

for i=0:4

movieConstruction(1, 'director\_nom') = {i};

critical\_test\_noms = predict(modelCritic, movieConstruction)

end

%%

%This section is strictly just to do some sensitivity analysis for critical

%Specifically, here, we are going to randomize the initial vector for a

%sample movie and then compute the metacritic prediction for all content

%ratings

fprintf('Dogdogogogogogogogogogog')

%This is for ADVENTURE fixed with content rating changing:

content\_ratings = {'PG\_13', 'PG', 'R', 'G', 'NC\_17', 'NR'};

set\_size = length(content\_ratings);

critical\_test = predict(modelCritic, movieConstruction)

for i=2:set\_size

movieConstruction(1, content\_ratings{i-1}) = {0};

movieConstruction(1, content\_ratings{i}) = {1};

critical\_test = predict(modelCritic, movieConstruction)

end

%%

%Iterate over this feature set:

for i=1:num\_features

%Get the current feature

feature = feature\_order(i);

%Retrieve the LB, UB, and step size for this feature

[LB, UB, step] = getIterParams(feature);

%For this feature, optimize it

optimal\_feature = optimizeFeature(feature, LB, UB, step, modelLow, modelMedium, modelHigh, modelCritic, movieConstruction, objFunctionParams);

%Now that we've optimized this feature, update the movie vector

%This is what will be fed into optimizeFeature in the next iteration:

movieConstruction(1, feature) = {optimal\_feature};

end

%Next, optimize for content rating (enforce it to only select one):

optimal\_content\_rating = optimizeContentRating(modelLow, modelMedium, modelHigh, modelCritic, movieConstruction, objFunctionParams);

content\_rating\_set = {'PG\_13', 'PG', 'R', 'G', 'NC\_17', 'NR'};

num = length(content\_rating\_set);

for i=1:num

current\_feature = content\_rating\_set{i};

if strcmp(current\_feature, optimal\_content\_rating) == 1

movieConstruction(1, current\_feature) = {1};

else

movieConstruction(1, current\_feature) = {0};

end

end

%Next, optimize for genre (enforce it to only select one:

optimal\_genre = optimizeGenre(modelLow, modelMedium, modelHigh, modelCritic, movieConstruction, objFunctionParams);

genre\_set = {'Adventure', 'Action', 'Animation', 'Family',...

'ScienceFiction', 'Comedy', 'Drama', 'Romance', 'Horror', 'Mystery',...

'Documentary', 'Music', 'Thriller', 'Crime', 'War'};

num = length(genre\_set);

for i=1:num

current\_feature = genre\_set{i};

if strcmp(current\_feature, optimal\_genre) == 1

movieConstruction(1, current\_feature) = {1};

else

movieConstruction(1, current\_feature) = {0};

end

end

%Next, optimize for day of week (for release):

optimal\_day = optimizeDay(modelLow, modelMedium, modelHigh, modelCritic, movieConstruction, objFunctionParams);

day\_set = {'Mon', 'Tue', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun'};

num = length(day\_set);

for i=1:num

current\_feature = day\_set{i};

if strcmp(current\_feature, optimal\_day) == 1

movieConstruction(1, current\_feature) = {1};

else

movieConstruction(1, current\_feature) = {0};

end

end

%Next, optimize for month of release:

optimal\_month = optimizeMonth(modelLow, modelMedium, modelHigh, modelCritic, movieConstruction, objFunctionParams);

month\_set = {'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',...

'Nov', 'Dec'};

num = length(month\_set);

for i=1:num

current\_feature = month\_set{i};

if strcmp(current\_feature, optimal\_month) == 1

movieConstruction(1, current\_feature) = {1};

else

movieConstruction(1, current\_feature) = {0};

end

end

fprintf('Optimization Complete!')

%%

%Output the predicted box office and metacritic score for this new movie:

critical\_output = predict(modelCritic, movieConstruction)

if movieConstruction{1, 'AdjustedBudget'} < 11

box\_office\_output = predict(modelLow, movieConstruction);

elseif movieConstruction{1, 'AdjustedBudget'} >= 11 && movieConstruction{1, 'AdjustedBudget'} < 75

box\_office\_output = predict(modelMedium, movieConstruction);

else

box\_office\_output = predict(modelHigh, movieConstruction);

end

true\_box\_office = exp(box\_office\_output)

%%

[title1, title2, title3, title4, title5] = findSimilarMovies(movieConstruction, movieDataAll,titles)

%So...at the end of this loop, movieConstruction should hold the optimal

%Return movies that "similar" (i.e. close in distance) to the output of

%this optimal movie....that could also just be a separate script (or

%actually, might be best to make that a function...pass in a movie vector

%and a data set and it'll go to town)

%findSimilarMovies(movieVector);

%^^this guy should technically iterate through all of the movies in our

%data set...and compute a similarity score for each...and spit out closest

fprintf("Done!!!...")

%%