function [title1, title2, title3, title4, title5] = findSimilarMovies(movie, allMovies, titles)

%Returns similar movies

% Takes a movie vector and returns (in decreasing order), the 3 movies in

% our data set that are most similar to it

%Optimization Start Time = 5:49pm

%Given a movie vector, we want to return the top 3 most s

%We will compute a similarity score for the continuous variables and then

%also a separate one for binary variables...and then combine them

%Declare the set of continuous variables:

cont\_vars = {'Num\_Theatres\_Widest', 'AdjustedBudget', 'sentiment', ...

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

'totNumWins', 'director\_nom', 'director\_win', 'total\_language'};

binary\_vars = {'Franchise\_Raw', 'PG\_13', 'PG', 'R', 'G', 'NC\_17', 'NR',...

'Adventure', 'Action', 'Animation', 'Family',...

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

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

%For the continuous variables, here are vectors containing the mean and

%stdev for each variable (in sequence):

cont\_means = [2002.36 58.42 -0.059 108.99 3.13 1.12 0.38 1.68 0.4 0.13 0.03 1.51];

cont\_stdev = [163.11 57.88 0.64 18.60 6.21 1.32 0.67 2.28 0.733 0.48 0.19 0.92];

numCont = length(cont\_vars);

%Now, what we're going to do is get a normalized vector for the movie of

%interest (the one that will act as a reference point)

normalized\_vec = zeros(1, numCont);

for i=1:numCont

current\_feature = cont\_vars{i};

current\_val = movie{1, current\_feature};

normalized\_val = (current\_val - cont\_means(i))/cont\_stdev(i);

normalized\_vec(i) = normalized\_val;

end

%Now, we need to iterate through of the movies

numMovies = 1714;

euclidean\_distances = zeros(1, numMovies);

for i=1:numMovies

raw\_values = zeros(1, numCont);

%Get the raw values now from the current movie under consideration

for j=1:numCont

current\_feature = cont\_vars{j};

current\_val = allMovies{i, current\_feature};

raw\_values(j) = current\_val;

end

%Now, this vector of raw values is ready to be normalized:

current\_normalized = zeros(1, numCont);

for j=1:numCont

current\_norm\_val = (raw\_values(j) - cont\_means(j))/cont\_stdev(j);

current\_normalized(j) = current\_norm\_val;

end

%Next, find the Euclidean distance between them....

total\_sum\_squares = 0;

for j=1:numCont

current\_difference = (normalized\_vec(j) - current\_normalized(j))^2;

total\_sum\_squares = total\_sum\_squares + current\_difference;

end

%Compute the final Euclidean distance:

current\_euclidean = sqrt(total\_sum\_squares);

%Add to the vector:

euclidean\_distances(i) = current\_euclidean;

end

%Find the 3 shortest distances:

%#1 Closest:

index1 = find(euclidean\_distances==min(euclidean\_distances));

%Make this index infinity:

euclidean\_distances(index1) = Inf;

%2nd Closest:

index2 = find(euclidean\_distances==min(euclidean\_distances));

%Make this index infinity:

euclidean\_distances(index2) = Inf;

%3rd Closest:

index3 = find(euclidean\_distances==min(euclidean\_distances));

euclidean\_distances(index3) = Inf;

index4 = find(euclidean\_distances==min(euclidean\_distances));

euclidean\_distances(index4) = Inf;

index5 = find(euclidean\_distances==min(euclidean\_distances));

title1 = titles(index1);

title2 = titles(index2);

title3 = titles(index3);

title4 = titles(index4);

title5 = titles(index5);

end