Annotated Code:

**unit GlobalValues;**

**//**Any variables have been required to be globally accessible.

interface

var

biasmask1: array [0 .. 48] of real;

biasmask2: array [0 .. 20] of real;

biasmask3: array [0 .. 2] of real;

outputvalue1,outputvalue2,outputvalue3:real;

SongId1:string;

SongId2,SongId3: string;

playsongid2,playSongId3: string;

recallflag:boolean;

implementation

end.

**unit MainMenu;**

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,

REST.Client,

System.Rtti, System.Bindings.Outputs, Vcl.Bind.Editors, Data.Bind.EngExt,

Vcl.Bind.DBEngExt, Data.DB, Data.Win.ADODB,

REST.Authenticator.OAuth,

GlobalValues, ResultsPage, REST.Types, Data.Bind.Components,

Data.Bind.ObjectScope, IPPeerClient, Vcl.WinXCtrls;

type

TfrmMainMenu = class(TForm)

lblTitle: TLabel;

shpEntryBackGroundLeft: TShape;

shpEntryBackhroundRight: TShape;

edtSongEntry1: TEdit;

edtSongEntry3: TEdit;

edtSongEntry2: TEdit;

edtCodeEntry: TEdit;

lblInstructionsRight1: TLabel;

lblInstructionsLeft1: TLabel;

Label1: TLabel;

lblInstructionsLeft2: TLabel;

lblInstructionsLeft3: TLabel;

Label2: TLabel;

Label3: TLabel;

RESTClient: TRESTClient;

RESTIdRequest: TRESTRequest;

RESTResponse: TRESTResponse;

btnPriorUse: TButton;

btnSongsSearch: TButton;

btnSettings: TButton;

OAuth2Authenticator1: TOAuth2Authenticator;

tblSongProperties: TADOTable;

RESTPropertiesRequest: TRESTRequest;

tblChosenSongs: TADOTable;

Label4: TLabel;

tblRecommendedSongs: TADOTable;

procedure btnSongsSearchClick(Sender: TObject);

procedure btnSettingsClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormShow(Sender: TObject);

procedure btnPriorUseClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

backgroundprocess1 = class(TThread)

// Runs a background process to prevent the neuralnetwork training crashing the program

private

public

procedure Execute(); override;

end;

var

frmMainMenu: TfrmMainMenu;

implementation

uses

Winapi.ShellApi, SettingsPage, neuralnetwork;

{$R \*.dfm}

procedure backgroundprocess1.Execute();

// Runs parallel to the main processing of the main menu

begin

neuralnetwork.train(SongID1, SongId2, SongId3);

frmMainMenu.tblChosenSongs.Close;

frmResultsPage.Show;

frmMainMenu.Hide;

end;

procedure TfrmMainMenu.btnSongsSearchClick(Sender: TObject);

var

i, k: Integer;

position, j: Integer;

dbsendtempreal: real;

flag: boolean;

Song2, Song1, Song3, response, dbsendtemp: string;

test: char;

properties, dbfields: array of string;

begin

properties := ['danceability', 'energy', 'speechiness', 'acousticness',

'valence', 'tempo', 'time\_signature'];

dbfields := ['Danceability', 'Energy', 'Speechiness', 'Acousticness',

'Valence', 'Tempo', 'Time Signature'];

Song1 := frmMainMenu.edtSongEntry1.Text; // More managable variable names

Song2 := frmMainMenu.edtSongEntry2.Text;

Song3 := frmMainMenu.edtSongEntry3.Text;

SongID1 := '';

SongId2 := '';

SongId3 := '';

// First Song

frmMainMenu.RESTIdRequest.Resource := '//search?q=' + Song1 +

'&type=track&limit=1';

// Sets up the search parameters

frmMainMenu.RESTIdRequest.Execute; // Runs the search

// The song ID is always followed by "is\_local"

// http://www.delphibasics.co.uk/RTL.asp?Name=ansipos

position := ansipos('is\_local', frmMainMenu.RESTResponse.Content);

// Finds the position of the id

response := frmMainMenu.RESTResponse.Content;

for i := 0 to 21 do // Adds each character of the id to the string.

begin

test := response[(position - 32 + i)];

// id starts 32 characters after the pos of 'is\_local'

SongID1 := SongID1 + test;

end;

// showmessage(SongId1);

frmMainMenu.RESTPropertiesRequest.Resource := 'audio-features/' + SongID1;

// Requests the audio features of the song.

frmMainMenu.RESTPropertiesRequest.Execute;

// End of Requests for song 1

// Send To database

frmMainMenu.tblSongProperties.Open;

if (frmMainMenu.tblSongProperties.Locate('ID', SongID1, []) = False) then

begin

frmMainMenu.tblSongProperties.Edit;

frmMainMenu.tblSongProperties.Insert;

// Only adds to song properties table if not already there

frmMainMenu.tblSongProperties.FieldByName('ID').AsString := SongID1;

frmMainMenu.tblSongProperties.FieldByName('Song Name').AsString := Song1;

for i := 0 to 5 do

begin

dbsendtemp := '';

response := frmMainMenu.RESTResponse.Content;

// showmessage(properties[i]);

position := ansipos(properties[i], response);

// showmessage(inttostr(position));

k := length(properties[i]);

j := 0;

repeat

dbsendtemp := dbsendtemp + response[position + j + k + 4];

j := j + 1;

until (response[position + j + k + 4] = ',') or

(response[position + j + k + 4] = '}');

// showmessage(dbsendtemp);

dbsendtempreal := strtofloat(dbsendtemp);

// showmessage(dbfields[i]);

if strtofloat(dbsendtemp) <> 0 then

begin

frmMainMenu.tblSongProperties.FieldByName(dbfields[i]).AsFloat :=

dbsendtempreal;

end

else

flag := True

end;

// showmessage(frmMainMenu.RESTResponse.Content);

if flag = False then

begin

try

frmMainMenu.tblSongProperties.Post;

// showmessage('song 1 success');

except

showmessage('Error Writing to the song properties table');

end;

end

else

showmessage('There is a problem with one of your search terms');

end;

frmMainMenu.tblSongProperties.Close;

flag := False;

// SECOND Song

frmMainMenu.RESTIdRequest.Resource := '//search?q=' + Song2 +

'&type=track&limit=1';

// Sets up the search parameters

frmMainMenu.RESTIdRequest.Execute; // Runs the search

// The song ID is always followed by "is\_local"

position := ansipos('is\_local', frmMainMenu.RESTResponse.Content);

// Finds the position of the id

response := frmMainMenu.RESTResponse.Content;

for i := 0 to 21 do // Adds each character of the id to the string.

begin

test := response[(position - 32 + i)];

// id starts 32 characters after the pos of 'is\_local'

SongId2 := SongId2 + test;

end;

// showmessage(SongId2);

frmMainMenu.RESTPropertiesRequest.Resource := 'audio-features/' + SongId2;

// Requests the audio features of the song.

frmMainMenu.RESTPropertiesRequest.Execute;

// End of Requests for song 1

// Send To database

frmMainMenu.tblSongProperties.Open;

if (frmMainMenu.tblSongProperties.Locate('ID', SongId2, []) = False) then

begin // Only adds to song properties table if not already there

frmMainMenu.tblSongProperties.Edit;

frmMainMenu.tblSongProperties.Insert;

frmMainMenu.tblSongProperties.FieldByName('ID').AsString := SongId2;

frmMainMenu.tblSongProperties.FieldByName('Song Name').AsString := Song2;

for i := 0 to 5 do

begin

dbsendtemp := '';

response := frmMainMenu.RESTResponse.Content;

// showmessage(properties[i]);

position := ansipos(properties[i], response);

// showmessage(inttostr(position));

k := length(properties[i]);

j := 0;

repeat

dbsendtemp := dbsendtemp + response[position + j + k + 4];

j := j + 1;

until (response[position + j + k + 4] = ',') or

(response[position + j + k + 4] = '}');

// showmessage(dbsendtemp);

dbsendtempreal := strtofloat(dbsendtemp);

// showmessage(dbfields[i]);

if strtofloat(dbsendtemp) <> 0 then

begin

frmMainMenu.tblSongProperties.FieldByName(dbfields[i]).AsFloat :=

dbsendtempreal;

end

else

flag := True

end;

// showmessage(frmMainMenu.RESTResponse.Content);

if flag = False then

begin

try

frmMainMenu.tblSongProperties.Post;

// showmessage('song 2 success');

except

showmessage('Error Writing to the song properties table');

end;

end

else

showmessage('There is a problem with one of your search terms');

end;

frmMainMenu.tblSongProperties.Close;

flag := False;

// THIRD Song

frmMainMenu.RESTIdRequest.Resource := '//search?q=' + Song3 +

'&type=track&limit=1';

// Sets up the search parameters

frmMainMenu.RESTIdRequest.Execute; // Runs the search

// The song ID is always followed by "is\_local"

// http://www.delphibasics.co.uk/RTL.asp?Name=ansipos

position := ansipos('is\_local', frmMainMenu.RESTResponse.Content);

// Finds the position of the id

response := frmMainMenu.RESTResponse.Content;

for i := 0 to 21 do // Adds each character of the id to the string.

begin

test := response[(position - 32 + i)];

// id starts 32 characters after the pos of 'is\_local'

SongId3 := SongId3 + test;

end;

// showmessage(SongID3);

frmMainMenu.RESTPropertiesRequest.Resource := 'audio-features/' + SongId3;

// Requests the audio features of the song.

frmMainMenu.RESTPropertiesRequest.Execute;

// End of Requests for song 2

// Send To database

frmMainMenu.tblSongProperties.Open;

if (frmMainMenu.tblSongProperties.Locate('ID', SongId3, []) = False) then

begin // Only adds to song properties table if not already there

frmMainMenu.tblSongProperties.Edit;

frmMainMenu.tblSongProperties.Insert;

// showmessage(frmMainMenu.RESTResponse.Content);

frmMainMenu.tblSongProperties.FieldByName('ID').AsString := SongId3;

frmMainMenu.tblSongProperties.FieldByName('Song Name').AsString := Song3;

for i := 0 to 5 do

begin

dbsendtemp := '';

response := frmMainMenu.RESTResponse.Content;

// showmessage(properties[i]);

position := ansipos(properties[i], response);

// showmessage(inttostr(position));

k := length(properties[i]);

j := 0;

repeat

dbsendtemp := dbsendtemp + response[position + j + k + 4];

j := j + 1;

until (response[position + j + k + 4] = ',') or

(response[position + j + k + 4] = '}');

// showmessage(dbsendtemp);

dbsendtempreal := strtofloat(dbsendtemp);

// showmessage(dbfields[i]);

if strtofloat(dbsendtemp) <> 0 then

begin

frmMainMenu.tblSongProperties.FieldByName(dbfields[i]).AsFloat :=

dbsendtempreal;

end

else

flag := True

end;

// showmessage(frmMainMenu.RESTResponse.Content);

if flag = False then

begin

try

frmMainMenu.tblSongProperties.Post;

// showmessage('song 3 success');

except

showmessage('Error Writing to the song properties table');

end;

end

else

showmessage('There is a problem with one of your search terms');

frmMainMenu.tblSongProperties.Close;

end;

flag := False;

frmMainMenu.tblChosenSongs.Open;

frmMainMenu.tblChosenSongs.Edit;

frmMainMenu.tblChosenSongs.Insert;

frmMainMenu.tblChosenSongs.FieldByName('Song1ID').AsString := SongID1;

frmMainMenu.tblChosenSongs.FieldByName('Song2ID').AsString := SongId2;

frmMainMenu.tblChosenSongs.FieldByName('Song3ID').AsString := SongId3;

try

frmMainMenu.tblChosenSongs.Post;

// Background thread to prevent crash

backgroundprocess1.Create(False);

except

showmessage('Error with the chosen songs table');

end;

end;

procedure TfrmMainMenu.btnPriorUseClick(Sender: TObject);

begin

try

strtoint(edtCodeEntry.Text);

recallflag := True;

//So the program knows that it should be recalling data on showing the page

frmResultsPage.Show;

frmMainMenu.Hide;

except

showmessage('This is not the correct format of code');

end;

end;

procedure TfrmMainMenu.btnSettingsClick(Sender: TObject);

begin

frmSettings.Show;

frmMainMenu.Hide;

end;

procedure TfrmMainMenu.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Application.Terminate;

end;

procedure TfrmMainMenu.FormCreate(Sender: TObject);

begin

tblChosenSongs.ConnectionString :=

'Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\james\Desktop\Project\ProjectDB.accdb;Persist Security Info=False';

tblSongProperties.ConnectionString :=

'Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\james\Desktop\Project\ProjectDB.accdb;Persist Security Info=False';

//Sets up the database connection string

end;

procedure TfrmMainMenu.FormShow(Sender: TObject);

begin

recallflag := False;

end;

end.

**unit neuralnetwork;**

// The mathmatical side of the program running all the calculations

// Lots of abstract numbers that have been tuned to produce the results

interface

procedure train(songID1, SongId2, SongId3: string);

function sigmoid(inputvalue: real): real;

function songtest(testsongId: string): real;

implementation

uses Mainmenu, Winapi.Windows, Winapi.Messages, System.SysUtils,

System.Variants,

System.Classes, Vcl.Graphics, dialogs, math, GlobalValues;

// ==============================================================================

function sigmoid(inputvalue: real): real;

// This function takes the input value and converts it into a decimal

// value in the range 0 to 1 no matter the input value (Logistic function)

begin

result := (1 / (1 + (Power(exp(1.0), (-inputvalue)))));

end;

// Will use this so normalise the values returned by neural networks

// 1 being a desired value and 0 suggesting no correlation

// ==============================================================================

// ==============================================================================

procedure train(songID1, SongId2, SongId3: string);

var

i, j, k, l, m, c: integer;

inputlayer, hiddenlayer1: array [0 .. 6] of real;

hiddenlayer2: array [0 .. 2] of real;

trendflags1: array [0 .. 48] of char;

// Does changing this value have a positive or negative effect #

changefoundflag: boolean;

trendflags2: array [0 .. 20] of char;

compoutputvalue1, compoutputvalue2, compoutputvalue3: real;

dbfields: array of string;

begin

dbfields := ['Danceability', 'Energy', 'Speechiness', 'Acousticness',

'Valence', 'Tempo', 'Time Signature']; // For iterating through field names

frmMainmenu.tblSongProperties.Active := False;

// Refresh Table to make sure it has up to date data

frmMainmenu.tblSongProperties.Active := True;

Repeat

begin

Repeat

begin

Repeat

begin

frmMainmenu.tblSongProperties.Locate('ID', songID1, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for i := 0 to 34 do // Initialise bias mask 1 with random function

begin

Randomize;

biasmask1[i] := (randomrange(-100000, 100000) / 500000);

// Generates a random number

// between -1 and 1

end;

for i := 35 to 41 do

begin

Randomize;

biasmask1[i] := (randomrange(-100000, 100000) / 30000000);

// Generated bias for this value accomodates for the fact that the tempo

// is around 200x larger than other values

end;

for i := 42 to 48 do // Initialise bias mask 1 with random function

begin

Randomize;

biasmask1[i] := (randomrange(-100000, 100000) / 500000);

// Generates a random number

// between -1 and 1

end;

for i := 0 to 20 do // Initialise bias mask 2 with random function

begin

Randomize;

biasmask2[i] := (randomrange(-100000, 100000) / 250000);

// Generates a random number

// between -1 and 1

end;

for i := 0 to 2 do // Initialise bias mask 3 with random function

begin

Randomize;

biasmask3[i] := (randomrange(-100000, 100000) / 250000);

// Generates a random number

// between -1 and 1

end;

// ------------------------------------------------------------------------------

// ---Song1-------

// Wipes hidden layer values

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

outputvalue1 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

outputvalue1 := 0;

for l := 0 to 2 do

begin

outputvalue1 := outputvalue1 + (hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song2-------

frmMainmenu.tblSongProperties.Locate('ID', SongId2, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

outputvalue2 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

outputvalue2 := outputvalue2 + (hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song3-------

frmMainmenu.tblSongProperties.Locate('ID', SongId3, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

outputvalue3 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

outputvalue3 := outputvalue3 + (hiddenlayer2[l] \* biasmask3[l]);

end;

//

end;

Until (sigmoid(outputvalue1) > sigmoid(outputvalue3)) and

(sigmoid(outputvalue2) > sigmoid(outputvalue3));

// ------------------------------------------------------------------------------

// Run CHecks for the first layer

for c := 0 to 48 do

begin

biasmask1[c] := biasmask1[c] + (0.05);

// Make a change to each bias and see how it affects outputs

// Runs values through again for each song:

frmMainmenu.tblSongProperties.Locate('ID', songID1, []);

// ------------------------------------------------------------------------------

// ---Song1-------

// Wipes hidden layer values

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue1 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue1 := compoutputvalue1 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song2-------

frmMainmenu.tblSongProperties.Locate('ID', SongId2, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue2 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue2 := compoutputvalue2 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song3-------

frmMainmenu.tblSongProperties.Locate('ID', SongId3, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue3 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue3 := compoutputvalue3 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

if ((compoutputvalue1 - outputvalue1) > 0) or

((compoutputvalue2 - outputvalue2) > 0) and

((compoutputvalue3 - outputvalue3) < 0) then

// does shifting this value up have a positive change

begin

trendflags1[c] := '+';

changefoundflag := True;

end

else

trendflags1[c] := 'n';

biasmask2[c] := biasmask2[c] - (0.05);

// Sets bias back to original value

// ------------------------------------------------------------------------------

// Try the reducing the bias

biasmask1[c] := biasmask1[c] - (0.05);

// Make a change to each bias and see how it affects outputs

// Runs values through again for each song:

frmMainmenu.tblSongProperties.Locate('ID', songID1, []);

// ------------------------------------------------------------------------------

// ---Song1-------

// Wipes hidden layer values

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue1 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue1 := compoutputvalue1 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song2-------

frmMainmenu.tblSongProperties.Locate('ID', SongId2, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue2 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue2 := compoutputvalue2 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song3-------

frmMainmenu.tblSongProperties.Locate('ID', SongId3, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue3 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue3 := compoutputvalue3 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

if ((compoutputvalue1 - outputvalue1) > 0) or

((compoutputvalue2 - outputvalue2) > 0) and

((compoutputvalue3 - outputvalue3) < 0) then

// does shifting this value up have a positive change

begin

trendflags1[c] := '-';

changefoundflag := True;

end

else

trendflags1[c] := 'n';

biasmask2[c] := biasmask2[c] + (0.05)

// Sets bias back to original value

end;

// ------------------------------------------------------------------------------

// Run CHecks for the second layer

for c := 0 to 20 do

begin

biasmask2[c] := biasmask2[c] + (0.05);

// Make a change to each bias and see how it affects outputs

// Runs values through again for each song:

frmMainmenu.tblSongProperties.Locate('ID', songID1, []);

// ------------------------------------------------------------------------------

// ---Song1-------

// Wipes hidden layer values

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue1 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue1 := compoutputvalue1 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song2-------

frmMainmenu.tblSongProperties.Locate('ID', SongId2, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue2 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue2 := compoutputvalue2 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song3-------

frmMainmenu.tblSongProperties.Locate('ID', SongId3, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue3 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue3 := compoutputvalue3 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

if ((compoutputvalue1 - outputvalue1) > 0) or

((compoutputvalue2 - outputvalue2) > 0) and

((compoutputvalue3 - outputvalue3) < 0) then

// does shifting this value up have a positive change

begin

trendflags2[c] := '+';

changefoundflag := True;

end

else

trendflags2[c] := 'n';

biasmask2[c] := biasmask2[c] - (0.05);

// Setting the bias back to original

// Try the reducing the bias

biasmask2[c] := biasmask2[c] - (0.05);

// Make a change to each bias and see how it affects outputs

// Runs values through again for each song:

frmMainmenu.tblSongProperties.Locate('ID', songID1, []);

// ------------------------------------------------------------------------------

// ---Song1-------

// Wipes hidden layer values

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue1 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue1 := compoutputvalue1 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song2-------

frmMainmenu.tblSongProperties.Locate('ID', SongId2, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue2 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue2 := compoutputvalue2 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song3-------

frmMainmenu.tblSongProperties.Locate('ID', SongId3, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

compoutputvalue3 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] +

(inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

compoutputvalue3 := compoutputvalue3 +

(hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

if ((compoutputvalue1 - outputvalue1) > 0) or

((compoutputvalue2 - outputvalue2) > 0) and

((compoutputvalue3 - outputvalue3) < 0) then

// does shifting this value up have a positive change

begin

trendflags2[c] := '-';

changefoundflag := True;

end

else

trendflags2[c] := 'n';

biasmask2[c] := biasmask2[c] + (0.05)

end;

// ------------------------------------------------------------------------------

end;

Until changefoundflag = True;

// Now apply the changes found to work out well

// ------------------------------------------------------------------------------

for c := 0 to 48 do

begin

if trendflags1[c] = '+' then

begin

biasmask1[c] := biasmask1[c] + (0.05)

end;

if trendflags1[c] = '-' then

begin

biasmask1[c] := biasmask1[c] - (0.05)

end;

end;

// ------------------------------------------------------------------------------

for c := 0 to 20 do

begin

if trendflags2[c] = '+' then

begin

biasmask2[c] := biasmask2[c] + (0.05)

end;

if trendflags2[c] = '-' then

begin

biasmask2[c] := biasmask2[c] - (0.05)

end;

end;

// ------------------------------------------------------------------------------

// Final run to get baseline values that the network gives:

// ------------------------------------------------------------------------------

// ---Song1-------

// Wipes hidden layer values

frmMainmenu.tblSongProperties.Locate('ID', songID1, []);

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

outputvalue1 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] + (inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

outputvalue1 := outputvalue1 + (hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song2-------

frmMainmenu.tblSongProperties.Locate('ID', SongId2, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

outputvalue2 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] + (inputlayer[k] \* biasmask1[7 \* j]);

end;

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

end;

for l := 0 to 2 do

begin

outputvalue2 := outputvalue2 + (hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

// ---Song3-------

frmMainmenu.tblSongProperties.Locate('ID', SongId3, []);

// Initialisation

// Addresses only the correct song ID

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

outputvalue3 := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] + (inputlayer[k] \* biasmask1[7 \* j]);

end;

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] +

(hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

end;

for l := 0 to 2 do

begin

outputvalue3 := outputvalue3 + (hiddenlayer2[l] \* biasmask3[l]);

end;

// ------------------------------------------------------------------------------

end;

Until (sigmoid(outputvalue1) > sigmoid(outputvalue3)) and

(sigmoid(outputvalue2) > sigmoid(outputvalue3));

outputvalue1 := (sigmoid(outputvalue1));

outputvalue2 := (sigmoid(outputvalue2));

outputvalue3 := (sigmoid(outputvalue3));

// ==============================================================================

end;

function songtest(testsongId: string): real;

var

dbfields: array of string;

i, j, k, l: integer;

inputlayer, hiddenlayer1: array [0 .. 6] of real;

outputvalue: real;

hiddenlayer2: array [0 .. 2] of real;

confidence: real;

begin

// Feed song Id into trained network

frmMainmenu.tblSongProperties.Locate('ID', testsongId, []);

dbfields := ['Danceability', 'Energy', 'Speechiness', 'Acousticness',

'Valence', 'Tempo', 'Time Signature'];

for i := 0 to 6 do // Initialise the input layer with values from DB

begin

inputlayer[i] := frmMainmenu.tblSongProperties.FieldByName

(dbfields[i]).AsFloat;

end;

for j := 0 to 6 do

begin

hiddenlayer1[j] := 0

end;

for j := 0 to 2 do

begin

hiddenlayer2[j] := 0

end;

outputvalue := 0;

for j := 0 to 6 do

begin

for k := 0 to 6 do

begin

hiddenlayer1[j] := hiddenlayer1[j] + (inputlayer[k] \* biasmask1[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for j := 0 to 2 do

begin

for k := 0 to 6 do

begin

hiddenlayer2[j] := hiddenlayer2[j] + (hiddenlayer1[k] \* biasmask2[7 \* j]);

end;

// showmessage(floattostr(hiddenlayer1[j]));

end;

for l := 0 to 2 do

begin

outputvalue := outputvalue + (hiddenlayer2[l] \* biasmask3[l]);

end;

outputvalue := sigmoid(outputvalue);

//

// Confidence Calculation

//

if outputvalue <= (outputvalue3 + 0.01) then

confidence := 0;

if (outputvalue > outputvalue1) and (outputvalue > outputvalue2) then

begin

if outputvalue1 > outputvalue2 then

begin

confidence := 0.5 + (outputvalue - outputvalue1) / 2 \*

(outputvalue2 - outputvalue3);

if confidence > 1 then

confidence := 1;

end;

if outputvalue2 > outputvalue1 then

begin

confidence := 0.5 + (outputvalue - outputvalue2) / 2 \*

(outputvalue1 - outputvalue3);

if confidence > 1 then

confidence := 1;

end;

end;

if (outputvalue < outputvalue1) and (outputvalue > outputvalue2) then

begin

confidence := 0.2 + ((outputvalue1 - outputvalue) /

(outputvalue1 - outputvalue2))

end;

if (outputvalue < outputvalue2) and (outputvalue > outputvalue1) then

begin

confidence := 0.2 + ((outputvalue2 - outputvalue) /

(outputvalue2 - outputvalue1))

end;

result := confidence

end;

end.

**unit resultsPage;**

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Data.DB, Vcl.Grids,

Vcl.DBGrids, Data.Win.ADODB, neuralnetwork, Vcl.Imaging.pngimage,

Vcl.ExtCtrls, globalvalues;

type

TfrmResultsPage = class(TForm)

btnBackToMenu: TButton;

tblSongProperties: TADOTable;

DataSource1: TDataSource;

tblSongRecommendations: TADOTable;

grdProperties: TDBGrid;

lblCaption1: TLabel;

lblReturnCode: TLabel;

lblTitle: TLabel;

shpEntryBackGroundLeft: TShape;

lblSong1Name: TLabel;

lblSong2Name: TLabel;

lblSong3Name: TLabel;

Image1: TImage;

Image2: TImage;

Image3: TImage;

DBGrid1: TDBGrid;

DataSource2: TDataSource;

procedure btnBackToMenuClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormShow(Sender: TObject);

procedure Image1Click(Sender: TObject);

procedure Image2Click(Sender: TObject);

procedure Image3Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmResultsPage: TfrmResultsPage;

implementation

uses mainmenu, Winapi.ShellApi;

{$R \*.dfm}

procedure TfrmResultsPage.btnBackToMenuClick(Sender: TObject);

begin

frmMainmenu.Show;

frmResultsPage.hide;

end;

procedure TfrmResultsPage.FormClose(Sender: TObject; var Action: TCloseAction);

begin

application.Terminate;

end;

procedure TfrmResultsPage.FormShow(Sender: TObject);

var

topconfidenceids: array [1 .. 3] of string;

columnsTotal, i, recordstocheck: integer;

confidence: real;

topconfidencevalues: array [1 .. 3] of real;

topconfidencenames: array [1 .. 4] of string;

begin

if recallflag = False then

begin

grdProperties.Enabled := True;

columnsTotal := grdProperties.DataSource.DataSet.RecordCount;

recordstocheck := round(columnsTotal \* 0.88);

DataSource1.DataSet.First;

for i := 1 to recordstocheck do

begin

// showmessage(DataSource1.DataSet.FieldByName('ID').AsString);

confidence := neuralnetwork.songtest(DataSource1.DataSet.FieldByName('ID')

.AsString); // runs procedure to find confidence value

// The following goes through and finds the song IDs with the highest confidence

// and then saves them in an array

grdProperties.DataSource.DataSet.Next;

if confidence > topconfidencevalues[1] then

begin

topconfidencevalues[1] := confidence;

topconfidenceids[1] := (DataSource1.DataSet.FieldByName('ID').AsString);

topconfidencenames[1] := (DataSource1.DataSet.FieldByName('Song Name')

.AsString);

end

else if confidence > topconfidencevalues[2] then

begin

topconfidencevalues[2] := confidence;

topconfidenceids[2] := (DataSource1.DataSet.FieldByName('ID').AsString);

topconfidencenames[2] := (DataSource1.DataSet.FieldByName('Song Name')

.AsString);

end

else if confidence > topconfidencevalues[3] then

begin

topconfidencevalues[3] := confidence;

topconfidenceids[3] := (DataSource1.DataSet.FieldByName('ID').AsString);

topconfidencenames[3] := (DataSource1.DataSet.FieldByName('Song Name')

.AsString);

end;

// Above finds top 3 confidence values

end;

lblSong1Name.Caption := ('1: ' + topconfidencenames[1]);

// Updates the UI to feature recommended songs

lblSong2Name.Caption := ('2: ' + topconfidencenames[2]);

lblSong3Name.Caption := ('3: ' + topconfidencenames[3]);

tblSongRecommendations.Open;

tblSongRecommendations.Edit;

tblSongRecommendations.Insert;

tblSongRecommendations.FieldByName('Song1ID').AsString :=

topconfidenceids[1];

tblSongRecommendations.FieldByName('Song2ID').AsString :=

topconfidenceids[2];

tblSongRecommendations.FieldByName('Song3ID').AsString :=

topconfidenceids[3];

try

tblSongRecommendations.Post;

// Table for recalling users recommended songs

except

showmessage('Error with the recommended songs table');

end;

lblReturnCode.Caption := inttostr(tblSongRecommendations.RecordCount);

// updates the user to give a number to be able to recall their recommendations

tblSongRecommendations.Close;

globalvalues.SongId1 := topconfidenceids[1];

globalvalues.playSongId2 := topconfidenceids[2];

globalvalues.playSongId3 := topconfidenceids[3];

end

else // only runs when the user is recalling past results

begin

DataSource2.DataSet.First; // Locates the correct entry in the database

for i := 1 to strtoint(frmMainmenu.edtCodeEntry.Text) do

begin

DataSource2.DataSet.Next;

end;

topconfidencenames[1] := DataSource2.DataSet.FieldByName('Song1Id')

.AsString; // Gets the Id required

topconfidencenames[2] := DataSource2.DataSet.FieldByName('Song2Id')

.AsString; // Gets the Id required

topconfidencenames[3] := DataSource2.DataSet.FieldByName('Song3Id')

.AsString; // Gets the Id required

globalvalues.SongId1 := topconfidenceids[1];

globalvalues.playSongId2 := topconfidenceids[2];

globalvalues.playSongId3 := topconfidenceids[3];

lblReturnCode.Caption := frmMainmenu.edtCodeEntry.Text;

lblSong1Name.Caption := ('1: ' + topconfidencenames[1]);

// Updates the UI to feature recommended songs

lblSong2Name.Caption := ('2: ' + topconfidencenames[2]);

lblSong3Name.Caption := ('3: ' + topconfidencenames[3]);

// Sets up the play buttons to open the correct page on click

end;

end;

procedure TfrmResultsPage.Image1Click(Sender: TObject);

begin

ShellExecute(0, 'OPEN', PChar('https://open.spotify.com/embed/track/' +

globalvalues.SongId1), '', '', SW\_SHOWNORMAL);

// Opens the spotify page of the song

end;

procedure TfrmResultsPage.Image2Click(Sender: TObject);

begin

ShellExecute(0, 'OPEN', PChar('https://open.spotify.com/embed/track/' +

globalvalues.playSongId2), '', '', SW\_SHOWNORMAL);

// Opens the spotify page of the song

end;

procedure TfrmResultsPage.Image3Click(Sender: TObject);

begin

ShellExecute(0, 'OPEN', PChar('https://open.spotify.com/embed/track/' +

globalvalues.playSongId3), '', '', SW\_SHOWNORMAL);

// Opens the spotify page of the song

end;

end.

**unit SettingsPage;**

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TfrmSettings = class(TForm)

lblTitle: TLabel;

btnBackToMenu: TButton;

edtAccessCodeEntry: TEdit;

lblAccessCodeInstruct: TLabel;

btnAccessCodeSubmit: TButton;

procedure btnBackToMenuClick(Sender: TObject);

procedure OnShow(Sender: TObject);

procedure btnAccessCodeSubmitClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmSettings: TfrmSettings;

implementation

uses

MainMenu,Winapi.ShellApi;

{$R \*.dfm}

procedure TfrmSettings.btnAccessCodeSubmitClick(Sender: TObject);

begin

FrmMainMenu.OAuth2Authenticator1.AccessToken:= edtAccessCodeEntry.Text;

end;

procedure TfrmSettings.btnBackToMenuClick(Sender: TObject);

begin

frmSettings.hide;

frmMainMenu.Show;

end;

procedure TfrmSettings.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Application.Terminate;

end;

procedure TfrmSettings.OnShow(Sender: TObject);

begin

//Opens the webpage where the access token is located

ShellExecute(0, 'OPEN', PChar('https://accounts.spotify.com/authorize?client\_id=e7b2c8f2ecf04736be11b1cb493cdd9d&redirect\_uri=http://localhost:9090/&scope=user-read-private%20user-read-email&response\_type=token&state=123'), '', '', SW\_SHOWNORMAL);

end;

end.