```
/*
D. Piston
IST 707
Final Project Code

Notes about SAS:
 - In SAS there are only two data types: character and numeric
 - Within the data types, a format can be appied to adjust the look of the value, but does not change the actual value of the data
 - SAS comes with many Procedures that provide an easy way to conduct a data operation with parameters created by SAS. Many of the operations c
    - Procedures can create a data set with an "out" options
 - System variables are created with a %let statement. Anytime that statement is seen, a new system variable was created that can be called by
    - Eaxmple: %let newvar = dan. The variable newvar was set to "dan".
 - This code will be utilizing SAS Cloud Analytics Services (CAS) to analyze the data. The data is taken from the standard environment and load
    - https://blogs.sas.com/content/sgf/2018/08/13/intro-cas-sas-viya/

Project Notes:
 - The project uses two models:
    1. Random Forest
    2. Neural Network

- Both models have lift charts and ROC graphs associated with them.
*/

/*Runs SAS formats. Fucntions as a look up tables and provides a label based on the column.*/
/*First line creates a path to where the Formats program is located*/
filename formats filesrvc folderpath='/Data Management/_SAS Studio/Admin' filename='IVMF Formats.sas';
/*Include statement runs the file location at the file path above*/
%include formats;

/*Creates SAS library. A SAS Library is a folder where data is stored*/
libname Project '/mnt/ra_dm_dev/dmpiston';

/*Create dataset needed for model*/
/*Proc SQL is the SAS version of SQL syntax. Functions similary as Oracle or SQL Management Studio*/
/*Creates a SAS dataset called "model_prep", cleans variables and renames variables*/

/*If the data already exists then the data will not be created again*/
%if %sysfunc(exist(project.model_set)) %then %do;
%put Model set exists on file.;
%end;
%else %do;
proc sql;
    create table project.model_set as
    select contact__contact_id                                  as id
           /*course attirbutes*/
           ,onlinecoursename                                    as course_name
           ,pe_pi__o2o_model                                    as course_model
           /*military attributes*/
           ,rank_single                                         as rank
               /*Select military connection based on first choice if more than one was selected*/
           ,case when upcase(contact__military_connection) like 'ACTIVE DUTY%' then 'Active Duty'
               when upcase(contact__military_connection) like 'SERVICEMEMBER%' then 'Active Duty'
               when upcase(contact__military_connection) like 'SPOUSE%' then 'Spouse'
               when upcase(contact__military_connection) like 'MILITARY SPOUSE%' then 'Spouse'
               when upcase(contact__military_connection) like 'DEPENDANT%' then 'Spouse'
               when upcase(contact__military_connection) like 'VETERAN%' then 'Veteran'
               when upcase(contact__military_connection) like 'RETIREE%' then 'Veteran'
               when upcase(contact__military_connection) like 'NATIONAL%' then 'National Guard/Reserve'
               when upcase(contact__military_connection) like 'RESERVE%' then 'National Guard/Reserve'
               when upcase(contact__military_connection) like 'NONE%' then ''
               ELSE '' end                                      as military_connection
               /*Given a null value if the word Missing was found*/
           ,case when branch_clean  = 'Missing' then ''
               else branch_clean end                            as military_branch
           /*demographic attirbutes*/
               /*In SAS a missing numeric value is given a period (.). If a period is found, a missing value is put in its place.*/
           ,case when educat = . then ''
               /*The put function changes a numeric variable to a character variable based on the format look up*/
               else put(educat,o2o_educat.) end                 as education
           ,case when gender = . then ''
               else put(gender,o2o_gender.)  end                as gender
           ,case when 17 <= ageatenroll <= 24 then '18-24'
               when 25 <= ageatenroll <= 34 then '25-34'
               when 35 <= ageatenroll <= 44 then '35-44'
               when 45 <= ageatenroll <= 54 then '25-54'
               when ageatenroll >= 55 then '55+'
               else '' end                                      as age
           ,racecatall                                          as race
           ,case when workhxcat3 = . then ''
               else put(workhxcat3,o2o_workhxcatthree.) end     as current_work_status
           ,case when upcase(s.statecode) = upcase(c.Contact__Mailing_State_Province) then s.statecode
               when upcase(s2.statename) = upcase(c.Contact__Mailing_State_Province) then s2.statecode
               else '' end                                      as state
           /*Target variable*/
           ,case when pe__status = 'Graduated' then 1
```

```
                else 0 end                                      as graduated
    from o2odsprd.o2o_cohorts c
    /*Left join on a state look uptable to help clean the state variable above*/
    left join prgrmdw.ivmf_states s on upcase(s.statecode) = upcase(c.Contact__Mailing_State_Province)
    left join prgrmdw.ivmf_states s2 on upcase(s2.statename) = upcase(c.Contact__Mailing_State_Province)
    /*Selecting enrolled participants from 2017 an onward*/
    where year(pe_pi__start_date) ge 2017 and flagcenrollednew=1 and onlinecoursename  ne '';
quit;
%end;

/*SAS Prodcedure to show metadata on dataset*/
proc contents data=project.model_set;
run;

proc sql;
    /*Stores the variables names into new variables*/
    select name
    into: column1-
    from dictionary.columns
    where upcase(memname) = 'MODEL_SET' and upcase(libname) = 'PROJECT' and upcase(name) <> 'ID';
    /*Stores the number of columns in a variable*/
    select count(name)
    into: colstop trimmed
    from dictionary.columns
    where upcase(memname) = 'MODEL_SET' and upcase(libname) = 'PROJECT' and upcase(name) <> 'ID';
quit;

/*Proc freq is a SAS procedure that creates a frequency table of an attribute*/
/*Step 1: Creates a stored procedure with a column name as the input to produce a frequency table*/
%macro frequency(column);
proc freq data=project.model_set nlevels;
    table &column/missing nocum nopercent nocol ;
run;
%mend frequency;

/*Step 2: Creates a loop*/
%macro frequencyloop;
%local i;
%do i=1 %to &colstop;
    %frequency(&&column&i)
%end;
%mend frequencyloop;

/*Step 3: Runs the Proc Frequncy and loops through all of the variables in the model_prep dataset */
%frequencyloop

/*Start a SAS CAS (Cloud Analytics Services) session*/
cas mySession sessopts=(caslib=casuser timeout=1800 locale="en_US");
/*Create location to save data called casuser*/
libname casuser cas;
/*Load table into memory in the location created above*/
proc casutil;
    load data=project.model_set outcaslib="CASUSER"
    casout="model_set";
run;

/*Setting a variatey of variables needed for the analysis below*/
/* Create a CAS engine libref to save the output data sets */
%let caslibname = casuser;
%let sasdata          = project.model_set;
%let casdata          = casuser.model_set;
%let partitioned_data = casuser.model_partition;
/* Specify the data set inputs and target */
%let class_inputs    = age course_model course_name current_work_status education gender military_branch military_connection race rank state;
%let interval_inputs = ;
%let target          = graduated;

/*Explore the data*/
proc cardinality data=casuser.model_set outcard=casuser.data_card;
run;

/*Print highlighted variables for report*/
proc sql;
    select _varname_
        ,_type_
        ,_rlevel_
        ,_nmiss_
        ,_mfreqchr_
    from casuser.data_card;
quit;

/*Data partitioning*/
proc partition data=casuser.model_set partition samppct=70;
    by graduated;
    output out=casuser.model_partition copyvars=(_ALL_);
run;
```

```
/*********************************************************************/
/* Identify variables that explain variance in the target            */
/*********************************************************************/
/* Discriminant analysis for class target */
proc varreduce data=casuser.model_partition technique=discriminantanalysis;
    class &target &class_inputs.;
    reduce supervised &target=&class_inputs. &interval_inputs. /            maxeffects=8;
    ods output selectionsummary=summary;
run;

data out_iter (keep=Iteration VarExp Base Increment Parameter);
    set summary;
    Increment=dif(VarExp);
    if Increment=. then Increment=0;
    Base=VarExp - Increment;
run;

proc transpose data=out_iter out=out_iter_trans;
    by Iteration VarExp Parameter;
run;

proc sort data=out_iter_trans;
    label _NAME_='Group';
    by _NAME_;
run;

/* Variance explained by Iteration plot */
proc sgplot data=out_iter_trans;
    title "Variance Explained by Iteration";
    yaxis label="Variance Explained";
    vbar Iteration / response=COL1 group=_NAME_;
run;

/*********************************************************************/
/* Build a predictive model using Random Forest                      */
/*********************************************************************/

proc forest data=&caslibname..model_partition ntrees=50 numbin=20 minleafsize=5;
    input &interval_inputs. / level = interval;
    input &class_inputs.    / level = nominal;
    target &target          / level = nominal;
    partition rolevar=_partind_(train='1' validate='0');
    code file="/mnt/ra_dm_dev/dmpiston/IST 707 Project Random Forest.sas";
    ods output FitStatistics=fitstats;
run;

/*********************************************************************/
/* Score the data using the generated model                          */
/*********************************************************************/
data &caslibname.._scored_forest;
    set &caslibname..model_partition;
    %include "/mnt/ra_dm_dev/dmpiston/IST 707 Project Random Forest.sas";
run;

/* create data set from forest stats output */
data fitstats;
    set fitstats;
    label Trees     = 'Number of Trees';
    label MiscTrain   = 'Training';
    label MiscValid = 'Validation';
run;

/* plot misclassification as function of number of trees */
proc sgplot data=fitstats;
    title "Training vs Validation";
    series x=Trees y=MiscTrain;
    series x=Trees y=MiscValid/
           lineattrs=(pattern=shortdash thickness=2);
    yaxis label='Misclassification Rate';
run;
title;

/*********************************************************************/
/* Assess model performance                                          */
/*********************************************************************/
proc assess data=&caslibname.._scored_forest;
    input p_graduated1;
    target &target / level=nominal event='1';
    fitstat pvar=p_graduated0 / pevent='0';
    by _partind_;
    ods output fitstat  = forest_fitstat
               rocinfo  = forest_rocinfo
               liftinfo = forest_liftinfo;
run;
```

```
/*************************************************************************/
/* Analyze model using ROC and Lift charts                               */
/*************************************************************************/
ods graphics on;
proc format;
    value partindlbl 0 = 'Validation' 1 = 'Training';
run;

/* Construct a ROC chart */
title 'Random Forest ROC';
proc sgplot data=forest_rocinfo aspect=1;
    title "ROC Curve";
    xaxis label="False positive rate" values=(0 to 1 by 0.1);
    yaxis label="True positive rate"  values=(0 to 1 by 0.1);
    lineparm x=0 y=0 slope=1 / transparency=.7 LINEATTRS=(Pattern=34);
    series x=fpr y=sensitivity /group=_partind_;
    format _partind_ partindlbl.;
run; title;

/* Construct a Lift chart */
title 'Random Forest Lift';
proc sgplot data=forest_liftinfo;
    title "Lift Chart";
    xaxis label="Population Percentage";
    yaxis label="Lift";
    series x=depth y=lift /
           group=_partind_ markers markerattrs=(symbol=circlefilled);
    format _partind_ partindlbl.;
run; title;
ods graphics off;

/*************************************************************************/
/* NEURAL NETWORK predictive model                                       */
/*************************************************************************/
proc nnet data=&partitioned_data;
  target &target / level=nom;
/*   input &interval_inputs. / level=int; */
  input &class_inputs. / level=nom;
  hidden 2;
  train outmodel=casuser.nnet_model;
  partition rolevar=_partind_(train='1' validate='0');
  ods exclude OptIterHistory;
run;

/*************************************************************************/
/* Score the data using the generated NN model                           */
/*************************************************************************/
title 'Neural Network Model';
proc nnet data=&partitioned_data inmodel=casuser.nnet_model noprint;
  output out=casuser._scored_NN copyvars=(_ALL_);
run;

proc assess data=&caslibname.._scored_NN;
    input p_graduated1;
    target &target / level=nominal event='1';
    fitstat pvar=p_graduated0 / pevent='0';
    by _partind_;
    ods output fitstat  = nn_fitstat
               rocinfo  = nn_rocinfo
               liftinfo = nn_liftinfo;
run;

ods graphics on;
/* Construct a ROC chart */
title 'Neural Network ROC';
proc sgplot data=nn_rocinfo aspect=1;
    title "ROC Curve";
    xaxis label="False positive rate" values=(0 to 1 by 0.1);
    yaxis label="True positive rate"  values=(0 to 1 by 0.1);
    lineparm x=0 y=0 slope=1 / transparency=.7 LINEATTRS=(Pattern=34);
    series x=fpr y=sensitivity /group=_partind_;
    format _partind_ partindlbl.;
run; title;

/* Construct a Lift chart */
title 'Neural Network Lift';
proc sgplot data=nn_liftinfo;
    title "Lift Chart";
    xaxis label="Population Percentage";
    yaxis label="Lift";
    series x=depth y=lift /
           group=_partind_ markers markerattrs=(symbol=circlefilled);
    format _partind_ partindlbl.;
```