En este documento se encuentran todos los anexos referentes a los códigos de MATLAB de todas las funciones que resultaron de los modelos en cada una de las técnicas de Machine Learning. Adicional se entregan al final los códigos de las interfaces de usuarios con las cuales se implementaron los modelos finales del proyecto.

ANEXO 1.

Función Modelo 1 – Técnica Árbol de Decisión.

```
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
%       data.
%
%      validationAccuracy: A double containing the accuracy in percent. In
%       the app, the History list displays this overall accuracy score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
```

```
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 04-Apr-2021 14:39:19


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'CellLoad', 'TrafficR99', 'AvaCodesHSDPA', 'RxQualAchievable', 'VoiceErlangs',
'ActiveUsersHSPA', 'TTILoad'};
predictors = inputTable(:, predictorNames);
response = inputTable.DiaMod1;
isCategoricalPredictor = [false, false, false, false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationTree = fitctree(...
   predictors, ...
   response, ...
   'SplitCriterion', 'gdi', ...
   'MaxNumSplits', 100, ...
   'Surrogate', 'off', ...
   'ClassNames', categorical({'NOK'; 'OK'}));

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);t
treePredictFcn = @(x) predict(classificationTree, x);
trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));
% Add additional fields to the result struct
trainedClassifier.RequiredVariables = {'ActiveUsersHSPA', 'AvaCodesHSDPA', 'CellLoad',
'RxQualAchievable', 'TTILoad', 'TrafficR99', 'VoiceErlangs'};
trainedClassifier.ClassificationTree = classificationTree;
trainedClassifier.About = 'This struct is a trained model exported from Classification Learner
R2020a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit =
c.predictFcn(T) \nreplacing ''c'' with the name of the variable that is this struct, e.g.
''trainedModel''. \n \nThe table, T, must contain the variables returned by: \n  c.RequiredVariables
\nVariable formats (e.g. matrix/vector, datatype) must match the original training data.
\nAdditional variables are ignored. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appclassification_exportmodeltoworkspace'')">How to predict using an exported model</a>.');
```

```
% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'CellLoad', 'TrafficR99', 'AvaCodesHSDPA', 'RxQualAchievable', 'VoiceErlangs',
'ActiveUsersHSPA', 'TTILoad'};
predictors = inputTable(:, predictorNames);
response = inputTable.DiaMod1;
isCategoricalPredictor = [false, false, false, false, false, false, false];
% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationTree, 'KFold', 5);
% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);
% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
```

ANEXO 2.

Función Modelo 2 – Técnica Árbol de Decisión.

```
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
%       data.
%
%      validationAccuracy: A double containing the accuracy in percent. In
%       the app, the History list displays this overall accuracy score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
```

```
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 04-Apr-2021 18:04:21


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'CellLoad', 'TrafficR99', 'AvaCodesHSDPA', 'RxQualAchievable', 'VoiceErlangs',
'ActiveUsersHSPA', 'TTILoad'};
predictors = inputTable(:, predictorNames);
response = inputTable.Diagnostico;
isCategoricalPredictor = [false, false, false, false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationTree = fitctree(...
   predictors, ...
   response, ...
   'SplitCriterion', 'deviance', ...
   'MaxNumSplits', 131, ...
   'Surrogate', 'off', ...
   'ClassNames', categorical({'CAP'; 'CHECK'; 'COV'}));

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(classificationTree, x);
trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
```

```
trainedClassifier.RequiredVariables = {'ActiveUsersHSPA', 'AvaCodesHSDPA', 'CellLoad',
'RxQualAchievable', 'TTILoad', 'TrafficR99', 'VoiceErlangs'};
trainedClassifier.ClassificationTree = classificationTree;
trainedClassifier.About = 'This struct is a trained model exported from Classification Learner
R2020a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit =
c.predictFcn(T) \nreplacing ''c'' with the name of the variable that is this struct, e.g.
''trainedModel''. \n \nThe table, T, must contain the variables returned by: \n  c.RequiredVariables
\nVariable formats (e.g. matrix/vector, datatype) must match the original training data.
\nAdditional variables are ignored. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appclassification_exportmodeltoworkspace'')">How to predict using an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'CellLoad', 'TrafficR99', 'AvaCodesHSDPA', 'RxQualAchievable', 'VoiceErlangs',
'ActiveUsersHSPA', 'TTILoad'};
predictors = inputTable(:, predictorNames);
response = inputTable.Diagnostico;
isCategoricalPredictor = [false, false, false, false, false, false, false];
% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationTree, 'KFold', 15);
% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);
% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
```

ANEXO 3.

Función Modelo 1 – Técnica Red Neuronal.

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 04-Apr-2021 22:26:34.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = Qx7 matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
```

```
%   Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [17.59;0;1.0202;2;0;0;0];
x1_step1.gain =
[0.000331966185924302;0.0109529025191676;0.146790068183987;0.0909090909090909;0.0574
382538770821;0.0622727046281074;0.02];
x1_step1.ymin = -1;

% Layer 1
b1 = [2.2062551080266548098;-1.8299608544184087222;3.210465974783645926;-
0.96958092449379684741;0.16867795703894658543;2.6483918514907549557;-
3.7060411160349060467;1.6265684728165175077;2.4230794774102957234;-
3.5306366497257495318;3.2518376151905612481;1.1187994476022724566;-
1.8363839106802164469;1.5939857060842634873;2.4780063639344245274];
IW1_1 = [-1.215461950046394124 0.013330280489906752225 -0.23532459056629162486
0.077884899508803590029 0.69200884799827577787 0.73974557695161069493
1.0513544792798452665;0.2749449868963457777 0.030224275920737106449 -
1.2987314416303923359 -0.60087029542782088853 1.1446961156412012173
0.57388288880884641419 -0.67389493203184935854;-0.74254614930180373733 -
0.30372965076120589378 -1.0255527300662190537 -1.0037149129806468562
0.52773517826760074634 2.4477878021870624892 -
1.1446950715793031694;0.83923452209737081642 0.42085505471537426603
0.18019226360436743639 -0.87671449143948521066 1.3409700120684848734 -
0.45774786266697359283 0.94251558159577608631;-0.95162250457271568838
0.47069081535625006962 0.66452799594242928372 -0.30422445452367896657
1.0098741582770107694 -1.0893288942379169004 -1.1073669624083908314;-
0.78362428556578844763 0.19456297397180355446 1.9821628557450228225 -
2.6418247489335748845 1.4623924463280140174 4.7512079760395007355
3.7633998619396016316;2.1937856383008123018 0.508218649891035823
0.251583186947213433 -0.11125515721134673908 0.29485010396405864075 -
6.9495151134967185769 0.17062387891427688302;-1.3066972597946115986 -
0.47768885575873215599 -0.05364046307952334558 -0.59084169553719900758 -
0.64222954538825849369 3.5186170174132631594 -1.35079488243663004;-
1.9511300748048840958 0.14249999153256714957 0.15485594675937969233 -
0.83443946580232453147 -0.72584611289204437057 5.300126856523447394 -
0.10072098513575386547;1.8162752966067374771 -0.84184913198008759494
1.8716102546475372748 2.0436636160938070894 1.1404573964131143882 -
```

```
3.9773399676890370813 -0.5919838625749452854;-0.12148076446647723681
0.99408111846552515978 -0.18000711136480165853 -1.9687228433092065494
0.35574449141519071116 2.9174750256630868073
0.93801406460963543399;1.0046812942352159048 0.56603526160564399117
0.82963458072044671265 -0.72989103689709422529 0.15467681187288576106 -
1.1472521311075363215 0.43788692614131030645;-0.48988002669841473891 -
0.84880783315017160895 -0.52385413889331911008 -0.82439933042540403108
0.87754740197765090937 0.060758105927689223802 -
1.0307838019660513496;1.1153201531073035913 -0.65355267030939967476 -
0.067268793285220299105 0.15328450777191532373 0.19993665483707234731 -
1.303290091233318293 -1.074630060519156993 2;0.40766528310534183577
0.15653163544867382262 0.92480966045977708312 1.1734344981232245342
0.54078872050401349814 -0.70504040244616938971 -0.56243024342119274372];

% Layer 2
b2 = -0.22828629835067981713;
LW2_1 = [-0.60786700172583041368 0.71185299413810809721 -4.0009762578663226051 -
1.1788515957900644882 0.67511380604125892368 -5.759475512599291136
7.623566070516663018 -4.0007669694128003002 -6.1034959500542225186
5.8475565161354303712 -4.2334266281581669489 -0.017162388527549830852 -
0.35580809618743919343 0.014631677647876298667 -1.3600714327337939391];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
  X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
  Q = size(X{1},1); % samples/series
else
  Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS
```

```
    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = logsig_apply(repmat(b2,1,Q) + LW2_1*a1);

    % Output 1
    Y{1,ts} = a2;
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Positive Transfer Function
function a = logsig_apply(n,~)
a = 1 ./ (1 + exp(-n));
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end
```

ANEXO 4.

Función Modelo 2 – Técnica Red Neuronal.

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 17-Apr-2021 21:45:17.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = Qx7 matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = Qx3 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [80.01;10.2;1.0202;2;0.0034;0;0];
x1_step1.gain =
[0.00033544158368680 5;0.0116009280742459;0.165511671055485;0.0909090909090909;0.0574
43862984898;0.0622727046281074;0.02];
x1_step1.ymin = -1;

% Layer 1
b1 = [2.1952541830367855447;1.9097904462786787771;-
1.4972149750382268696;2.0859671716170078426;1.3164027810283971398;1.16137853560453
60775;-0.040270113108277508229;-0.12948113653260392986;-0.3821970141452998293;-
0.76054856806268145153;1.5861791298816676488;1.3779848481670884386;-
1.1484791485471745887;0.88353358339374021835;-
1.1899219495369111055;1.5912755459254213086;-2.5685881277329869476;-
1.5988165691992572981;2.0568083831865928701];
IW1_1 = [-0.1100861534023478 0.53657726389061899397 -0.12813932781222722723 -
0.864787335924331102 -1.3215073989671628496 -1.1755140642055159628
0.56288898930390018904;-0.52406414709786675044 0.49363884508630484493
0.92004445069589257944 0.096713710473743261242 -0.41200355908887215106 -
1.0308356217944092403 -1.3422976386569922536;1.2116206953263914947
```

0.45538974414318772466 -0.039846269032456659021 -0.97721449652683534115 -
0.76221335618599661554 -1.110189922097515991 -0.58262081608174676184;-
1.1421156122403706146 2.1906968503818764482 -1.5235905199758090767 -
0.99999433062086362511 0.026343300025971808087 1.2330750537487249474
1.9081815071897643143;-1.1659716354401588045 -0.11110766925683701611
0.60150703191506438294 -0.71623641424238726483 -0.87068949734310041588
1.0927935952501863781 0.52935797284400387674;-1.2252905147645920536
1.1428362966401917067 0.67080812040325576984 0.30631145721348784772 -
0.75042401711500994477 0.43009936855413610779
0.35526642131285796067;0.059526042421872642219 2.4218233284391219584 -
0.45065066044513329979 -4.6237000327257069543 1.9136538800594407039 -
0.65006761706639115683 2.6343914614485441916;0.94580398482205951805 -
1.1499374849206625004 1.0392217691514553657 0.16597008204219565464
0.18240796138528070358 0.54365244369250442791 1.0439175476008639709;-
1.0345393354174359235 0.031985238428458753757 1.2643907306084118947 -
4.1275129121592000203 -2.5897589365586437538 0.43600004829472494006 -
2.50786176288607221;0.66009340225318680062 0.34009625795950759741 -
0.07416907855678515904 0.17522780851938368762 -3.7990569216945133668
1.460227798400022925 1.2000564757803755533;-0.84960438699911511318
2.9161017861503943926 -1.0540995637755410375 -0.97658411108033915671
1.4407478773852562259 -0.68539280998334650707
0.79905572730430440842;1.4553779082339914908 -3.5861373455660570109
0.92749951618175052737 -1.3879840313863613943 2.3113744097163695379 -
1.4770268177643142149 -0.23304139414208979719;-0.69257105412008745926 -
5.7020830814132059672 0.57672233708006470643 -6.6105034210468129885
1.1932628827883031697 -0.019305772598556838332
0.63663567112286834071;0.80192434970416337503 -0.37405874702433483669 -
1.4232609581675141897 -1.9386770914832018864 -0.2942491293561756982 -
1.2115755992611845215 -0.80431000662587648353;-0.031837211424966718742 -
0.66473181485097665444 2.2602741173080467263 -1.5674290524217757703
1.8132342440779840587 -0.55189682060102274175
0.90307637606208301939;0.71066004880416966394 -0.68611046005087750199 -
1.0155571051382519254 -1.5870393332312322521 -0.16877494069151224854
1.1143718980885117809 -0.91417161388133605371;1.0671290208700479063 -
6.1782456872344990728 -1.0432297329095756577 1.0960906123616913632 -
1.6323643328742880598 0.44578135252724093718 -0.35942200864882362099;-
0.32511249121219731872 -2.9445789508107087507 -2.322105769376756141
0.77908833526271115133 1.5052060887373504006 -0.87587701909431914249 -
0.67611103684959705795;1.1205926519171598432 -0.57570220927069404304 -
0.2698968648723020225 0.19015329095569785678 0.99604408537451138983
0.19601349177795227008 1.561890893927915025];

% Layer 2
b2 = [1.0091565352373677023;-0.039806431816813418745;0.31122158128256849352];

```
LW2_1 = [1.1371589314644194868 0.11551995479260289412 -0.07299074938193994555 -
2.7083984439280337497 -0.21027027476657325633 1.0631398601905075463
4.4152204470028335592 0.76863342350829821914 -3.0494261704030067861 -
1.0855642235625500014 -1.5344149678529686032 1.1521842968779545391 -
3.4791284986888721953 0.11199155500458872481 2.2901516322790729241 -
0.4386965534420882995 3.0240738529753423691 -1.3461808397997914177
0.96718657198190416402;-0.16267905100856355771 -0.29059724209892756619
0.87170039839410728977 -0.82531244652166846354 -0.18051483521462216575 -
0.64055465241075359994 -0.63884737247142020511 0.18664069713087697022
0.86652875213838576052 -2.2990279725950646394 1.2418924865345064568
1.928591060964499837 -1.6943028835859390568 -0.45258100887535740453 -
0.87732716678380406883 1.5595458467212586218 0.20881212421287109748
3.5527437436493292111 -1.3922505103169915408;-0.82554505879272022106
0.16049712431583151862 0.093762691923820667972 0.94150024282610211834 -
0.92586823699580289571 -0.34352733961266435081 -4.5953797506955513796 -
0.3031002368597874308 3.1641501206581010841 1.3195850553691501617
1.5678004552991309239 -3.9633709429629351284 5.086094184339466473
0.048957587407776087529 -1.0836563720371039654 -1.5335555668390743467 -
3.6037517216115824148 -1.501324950322220575 -0.13322636516301544773];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
  X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
  Q = size(X{1},1); % samples/series
else
  Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

  % Input 1
  X{1,ts} = X{1,ts}';
```

```
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = softmax_apply(repmat(b2,1,Q) + LW2_1*a1);

    % Output 1
    Y{1,ts} = a2;
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Competitive Soft Transfer Function
function a = softmax_apply(n,~)
if isa(n,'gpuArray')
    a = iSoftmaxApplyGPU(n);
else
    a = iSoftmaxApplyCPU(n);
end
end
function a = iSoftmaxApplyCPU(n)
nmax = max(n,[],1);
n = bsxfun(@minus,n,nmax);
numerator = exp(n);
```

```
denominator = sum(numerator,1);
denominator(denominator == 0) = 1;
a = bsxfun(@rdivide,numerator,denominator);
end
function a = iSoftmaxApplyGPU(n)
nmax = max(n,[],1);
numerator = arrayfun(@iSoftmaxApplyGPUHelper1,n,nmax);
denominator = sum(numerator,1);
a = arrayfun(@iSoftmaxApplyGPUHelper2,numerator,denominator);
end
function numerator = iSoftmaxApplyGPUHelper1(n,nmax)
numerator = exp(n - nmax);
end
function a = iSoftmaxApplyGPUHelper2(numerator,denominator)
if (denominator == 0)
   a = numerator;
else
   a = numerator ./ denominator;
end
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end
```

ANEXO 5.

Función GUIDE 1 – Interfaz Gráfica de Usuario 1.

```
function varargout = MODUNO(varargin)
% MODUNO MATLAB code for MODUNO.fig
%      MODUNO, by itself, creates a new MODUNO or raises the existing
%      singleton*.
%
%      H = MODUNO returns the handle to a new MODUNO or the handle to
%      the existing singleton*.
%
%      MODUNO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MODUNO.M with the given input arguments.
%
%      MODUNO('Property','Value',...) creates a new MODUNO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
```

```
%     applied to the GUI before MODUNO_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to MODUNO_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MODUNO

% Last Modified by GUIDE v2.5 09-May-2021 07:05:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MODUNO_OpeningFcn, ...
                   'gui_OutputFcn',  @MODUNO_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MODUNO is made visible.
function MODUNO_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MODUNO (see VARARGIN)

% Choose default command line output for MODUNO
%NF=0;
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MODUNO wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MODUNO_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text2, 'BackgroundColor', 'w');
set(handles.text3, 'BackgroundColor', 'w');
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
T = get(handles.uitable1, 'Data');
B = cell2table(T);
C = table2array(B(:,2));
set(handles.text2, 'BackgroundColor', 'g');
set(handles.text3, 'BackgroundColor', 'g');
handles.m = 0;
%disp(C);
h = categorical({'OK','NOK'});
j = [0 0];
NF=size(T,1);
for i=1:NF
    if C(i,1)<80
      T(i,9) = cellstr("OK");
      j(1,1)=j(1,1)+1;
    else
      T(i,9) = cellstr("NOK");
      j(1,2)=j(1,2)+1;
      handles.m = handles.m + 1;
    end
end
set(handles.text4, 'String', num2str(j(1,2)));
set(handles.text5, 'String', num2str(j(1,1)));
```

```
set(handles.text2, 'BackgroundColor', 'w');
set(handles.text3, 'BackgroundColor', 'w');
bar(h,j);
set(handles.uitable1, 'Data', T);
guidata(hObject, handles);

% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Archivo_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Importar_Callback(hObject, eventdata, handles)
[nombre, direccion] = uigetfile({'*.xlsx','Archivo de Excel'}, 'Escoge un Archivo');
[M, string, full] = xlsread([direccion, nombre]);
NombresCol = full(1,:);
full(1,:)=[];
NFilas = 1:size(full,1);
numfilas = size(full,1);
numcolumnas = size(full,2);
set(handles.uitable1, 'Data', cell(numfilas, numcolumnas));
set(handles.uitable1, 'Data', full, 'ColumnName', NombresCol, 'RowName', NFilas);
guidata(hObject, handles);
% hObject    handle to Importar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Exportar_Callback(hObject, eventdata, handles)
[nombre, direccion] = uiputfile({'*xlsx', 'Archivo de Excel'}, 'Guardar Como');
Encabezado = get(handles.uitable1, 'ColumnName')';
Tabla = get(handles.uitable1, 'Data');
xlswrite([direccion, nombre], [Encabezado ; Tabla])
% hObject    handle to Exportar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
T = get(handles.uitable1, 'Data');
B = cell2table(T);
C = table2array(B(:,9));
handles.D = get(handles.uitable1, 'ColumnName');
E = cell(handles.m, 9);
a = handles.m;
b = a;
%disp(a);
%disp(D);
%disp(size(D));
NF=size(T,1);
for i=1:NF
    if C(i,1)=="NOK"
      c = (b-a)+1;
      a = a - 1;
      E(c,:) = T(i,:);
    end
end
%disp(E);
handles.MS=E;
guidata(hObject, handles);
MODDOS

% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

ANEXO 6.

Función GUIDE 2 – Interfaz Gráfica de Usuario 2.

```
function varargout = MODDOS(varargin)
% MODDOS MATLAB code for MODDOS.fig
%      MODDOS, by itself, creates a new MODDOS or raises the existing
%      singleton*.
%
%      H = MODDOS returns the handle to a new MODDOS or the handle to
%      the existing singleton*.
%
```

```
%      MODDOS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MODDOS.M with the given input arguments.
%
%      MODDOS('Property','Value',...) creates a new MODDOS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MODDOS_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MODDOS_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MODDOS

% Last Modified by GUIDE v2.5 09-May-2021 06:44:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MODDOS_OpeningFcn, ...
                   'gui_OutputFcn',  @MODDOS_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MODDOS is made visible.
function MODDOS_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to MODDOS (see VARARGIN)

mainHandles = guidata(MODUNO);
set(handles.uitable1, 'ColumnName', mainHandles.D);
set(handles.uitable1, 'Data', mainHandles.MS);
set(handles.text2, 'BackgroundColor', 'w');
set(handles.text3, 'BackgroundColor', 'w');
set(handles.text4, 'BackgroundColor', 'w');
% Choose default command line output for MODDOS
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MODDOS wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MODDOS_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
T = get(handles.uitable1, 'Data');
B = cell2table(T);
C2 = table2array(B(:,2));
C3 = table2array(B(:,3));
C4 = table2array(B(:,4));
C5 = table2array(B(:,5));
C6 = table2array(B(:,6));
%handles.m = 0;
set(handles.text2, 'BackgroundColor', 'g');
set(handles.text3, 'BackgroundColor', 'g');
set(handles.text4, 'BackgroundColor', 'g');
%disp(C2);
h = categorical({'COV','CAP', 'CHECK'});
```

```
j = [0 0 0];
NF=size(T,1);
for i=1:NF
  if C5(i,1)<17.5
    if C6(i,1)<10.1659
      if C6(i,1)<0.06594
      T(i,9) = cellstr("CHECK");
      j(1,3)=j(1,3)+1;
      else
      T(i,9) = cellstr("COV");
      j(1,1)=j(1,1)+1;
      end
    else
      if C3(i,1)<59.8
        if C4(i,1)<8.1244
        T(i,9) = cellstr("CHECK");
        j(1,3)=j(1,3)+1;
        else
        T(i,9) = cellstr("COV");
        j(1,1)=j(1,1)+1;
        end
      else
      T(i,9) = cellstr("CAP");
      j(1,2)=j(1,2)+1;
      end
    end
  else
    if C3(i,1)< 59.95
      if C4(i,1)<8.0001
        if C6(i,1)<10.4617
          if C6(i,1)<0.83609
          T(i,9) = cellstr("CHECK");
          j(1,3)=j(1,3)+1;
          else
          T(i,9) = cellstr("CAP");
          j(1,2)=j(1,2)+1;
          end
        else
        T(i,9) = cellstr("CHECK");
        j(1,3)=j(1,3)+1;
        end
      else
      T(i,9) = cellstr("CAP");
      j(1,2)=j(1,2)+1;
```

```
      end
    else
      if C6(i,1)<10.1284
      T(i,9) = cellstr("COV");
      j(1,1)=j(1,1)+1;
      else
        if C2(i,1)<376.085
        T(i,9) = cellstr("CAP");
        j(1,2)=j(1,2)+1;
        else
        T(i,9) = cellstr("CHECK");
        j(1,3)=j(1,3)+1;
        end
      end
    end
  end
end
%T(:,8)=num2cell(A(:,8));
bar(h,j);
set(handles.text7, 'String', num2str(j(1,1)));
set(handles.text6, 'String', num2str(j(1,3)));
set(handles.text5, 'String', num2str(j(1,2)));
set(handles.text2, 'BackgroundColor', 'w');
set(handles.text3, 'BackgroundColor', 'w');
set(handles.text4, 'BackgroundColor', 'w');
set(handles.uitable1, 'Data', T);
guidata(hObject, handles);
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Archivo_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Importar_Callback(hObject, eventdata, handles)
% hObject    handle to Importar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
function Exportar_Callback(hObject, eventdata, handles)
[nombre, direccion] = uiputfile({'*xlsx', 'Archivo de Excel'}, 'Guardar Como');
Encabezado = get(handles.uitable1, 'ColumnName')';
Tabla = get(handles.uitable1, 'Data');
xlswrite([direccion, nombre], [Encabezado ; Tabla])
% hObject    handle to Exportar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```