Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales

Máster en Automática y Robótica

Applied Artificial Intelligence

Assignment 3.1: k-nn and Bayesian Classifier

Josep María Barberá Civera (17048)

March 12, 2024

ASSIGNMENT 2.3

k-nn and Bayesian Classifier

Here are listed the main instructions for the task completion:

1. First load the synthetically created "data_D2_C2" data where the `p.value` dataset will be your training data and the `t.value` dataset will serve as the validation data.
2. Use the `nn-classifier` for computing the number of *miss-classifications* errors on `p.value` and on `t.value`, but remember to use in both cases `p.value` as the learning data.
3. Repeat again but now using the `Bayesian classifier`.
4. Compute the miss-classification error on `t.value` when using a Bayesian classifier with the following *a priori* probabilities: [1 0.1], [1 2] & [1 10].
5. Discuss the results.

1.1 Methodology

Thanks to the `fitcknn` and `fitcnb` functions, it is very easy to train the classifiers, parametric (`knn`) and non-parametric (`bayesian`) respectively.

On the other hand, once trained, the `predict` function, given a model and an input value, returns the estimate of the most probable class. Iterating over all the data of the training and validation sets, the behaviour of these algorithms has been evaluated and the results represented as confusion matrices (see next section (1.2)).

Finally, to better visualise the behaviour of the classifiers, a grid of points has been constructed that coincides with the domain of the data and the classification has been calculated for each node of this grid and the correct value of each point has been represented on it. It can then be seen where the classification fails and how the domains change depending on the initial value given to the `predict` function.

1.2 Discussion and Results

The figures obtained in the validation of the models are included below.

1.2.1 knn classifier

The k-nearest neighbour classifier is supervised and non-parametric, resulting in low generalisation with consequent noise learning. Figure 1.a shows the high specialisation of the model which for the data provided to learn the exact zones. While in Figure 1.b for the validation data, the classification fails in some points.

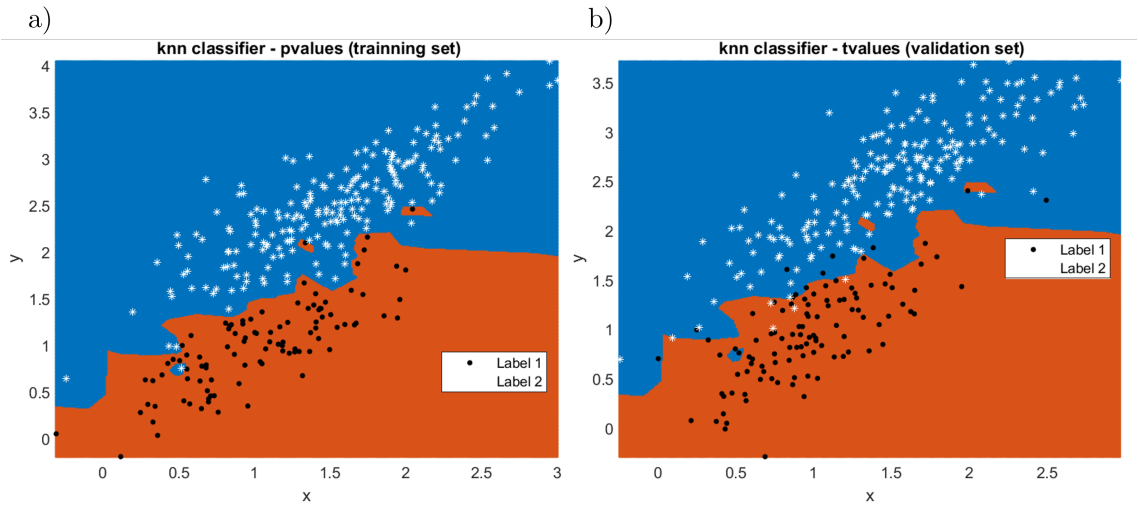

Figure 1: *Map of predictions using the* ***knn*** *model together with the correctly labelled points. In a) the training data while in b) the validation data.*

This means that the confusion matrix associated with the training data does not present any type of error, while for the validation data it does. It is seen in Figure 2.a for the `pvalues` data points and in Figure 2.b for the `tvalues` where there are about 15 erroneous points.

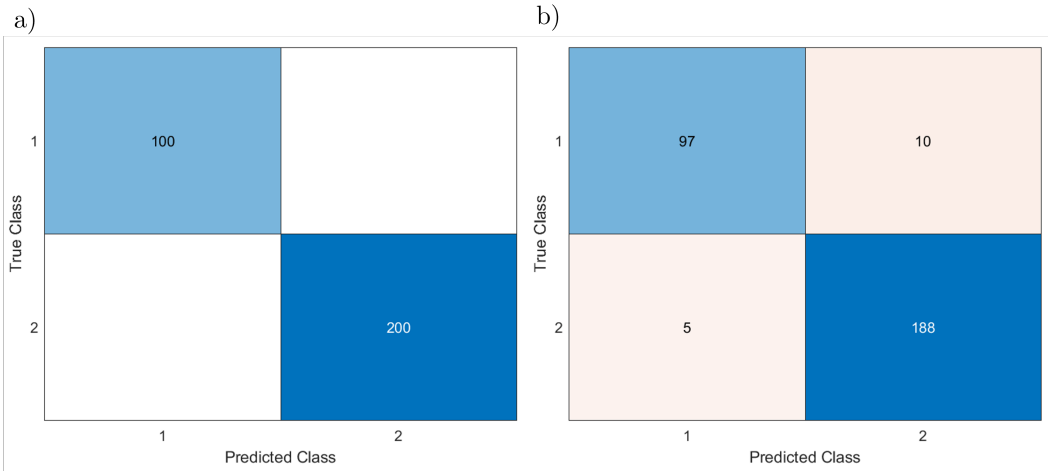

Figure 2: *Confusion matrix for the* ***knn*** *model. In a) for the same training values while in b) for the validation values.*

The accuracy associated with the confusion matrices are the following:

$$\mathrm{ACC}_1 = 100\,\%\,; \quad \mathrm{ACC}_2 = 95\,\%$$

1.2.2 Bayesian classifier

The `Bayesian classifier`, on the other hand, has an inherently generalising capacity as opposed to the data storage that the `knn classifier` used to perform. On the other hand, this model assumes that the data follow a normal distribution, which in our case is not true, leading to poor behaviour.

First, the results obtained for the Bayesian model without any prior estimation are shown. Thus, Figure 3 shows the frontier obtained against the points with their correct classification. It contrasts the smoothness of the model that presents little noise, but fails catastrophically in the classification. This occurs both for the training values themselves (Fig. 3.a) and for the validation data (Fig. 3.b).

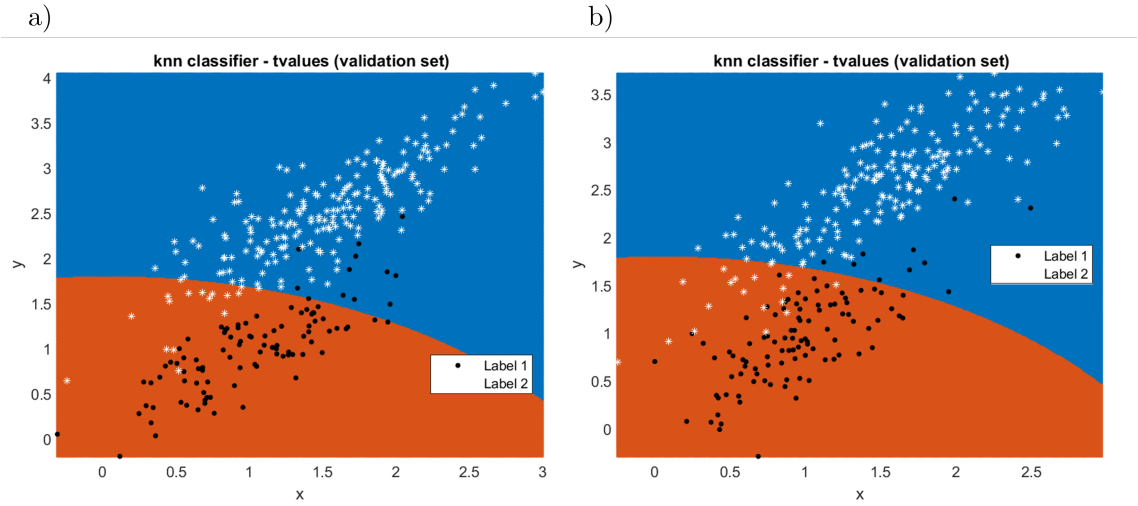

Figure 3: *Map of predictions using the* ***bayesian*** *model together with the correctly labelled points. In a) the training data while in b) the validation data.*

This is corroborated by the confusion matrices in Figure 4, since the model does not change its boundary either, both confusion matrices present similar values of accuracy, being in this case equal, although this need not always be the case.

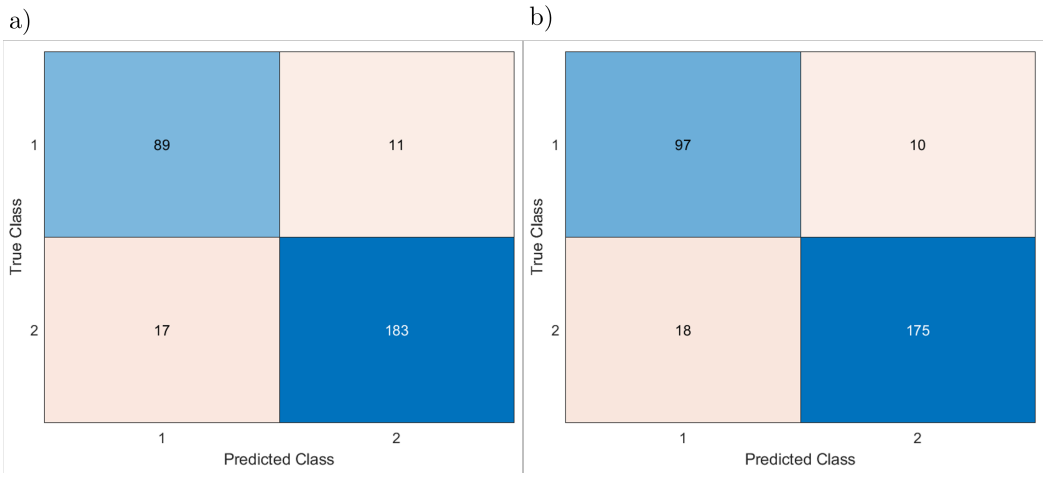

Figure 4: *Confusion matrix for the* ***bayesian*** *model. In a) for the same training values while in b) for the validation values.*

The accuracy associated with the confusion matrices are the following:

$$\mathrm{ACC}_1 = 90.67\,\%\,; \quad \mathrm{ACC}_2 = 90.67\,\%$$

The same process has been repeated for three different values of `initial probabilities` or `initial values` in the training of the model. These values are the next: $[1\ \ 0.1], [1\ \ 2]$ & $[1\ \ 10]$.

Figure 5 shows the evolution of the frontier for the three possible cases and the training data (right) or the validation data (left).

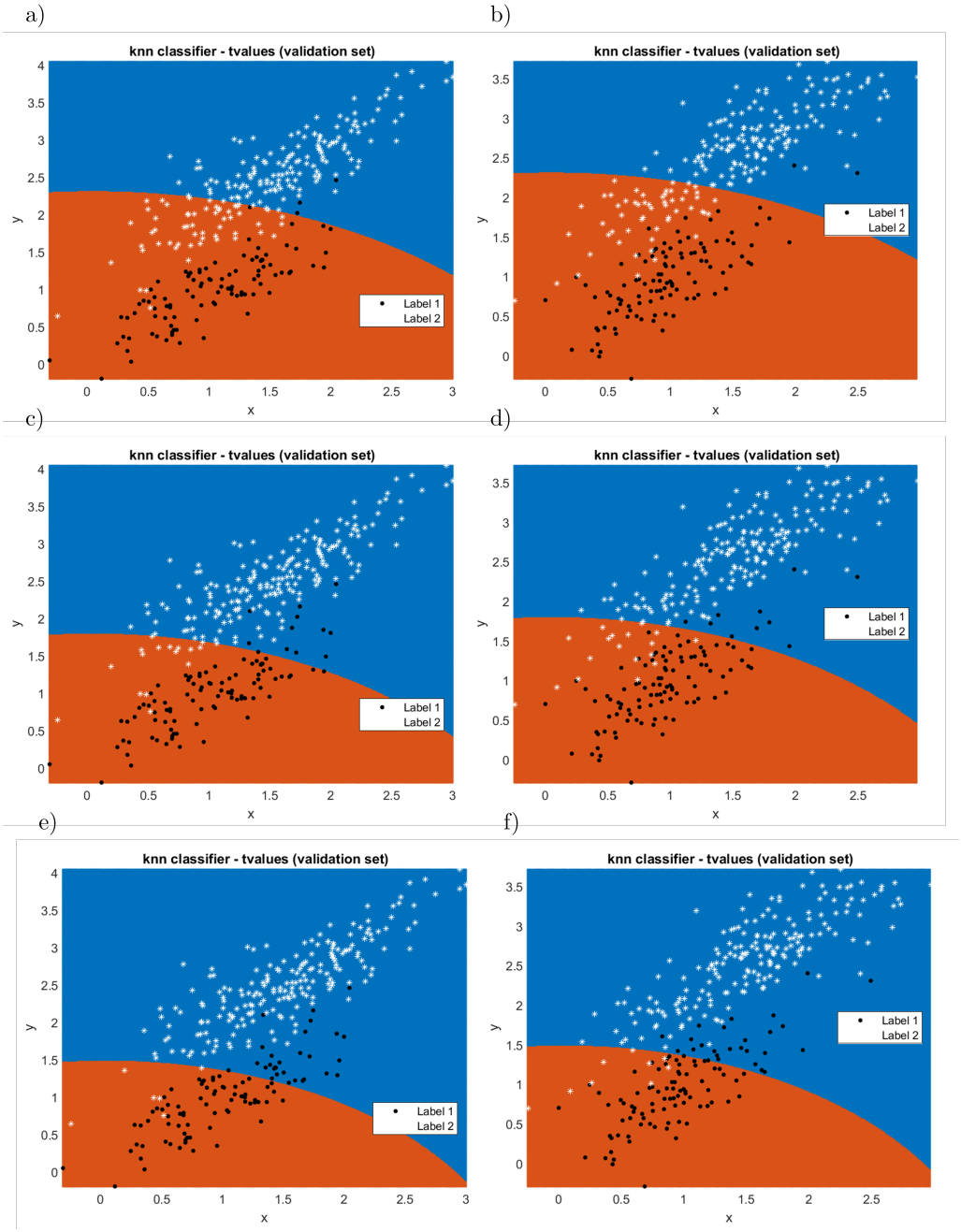

Figure 5: *Map of predictions using the* ***bayesian*** *model along with some prior probabilities values, together with the correctly labelled points. In a), c) and e) the training data while in b), d) and f) the validation data. In a) and b) the initial guess is* $[1\ \ 0.1]$*, in c) and d) is* $[1\ \ 2]$*, while in e) and f) is* $[1\ \ 10]$*.*

The associated confusion matrices are included in Figure 6.

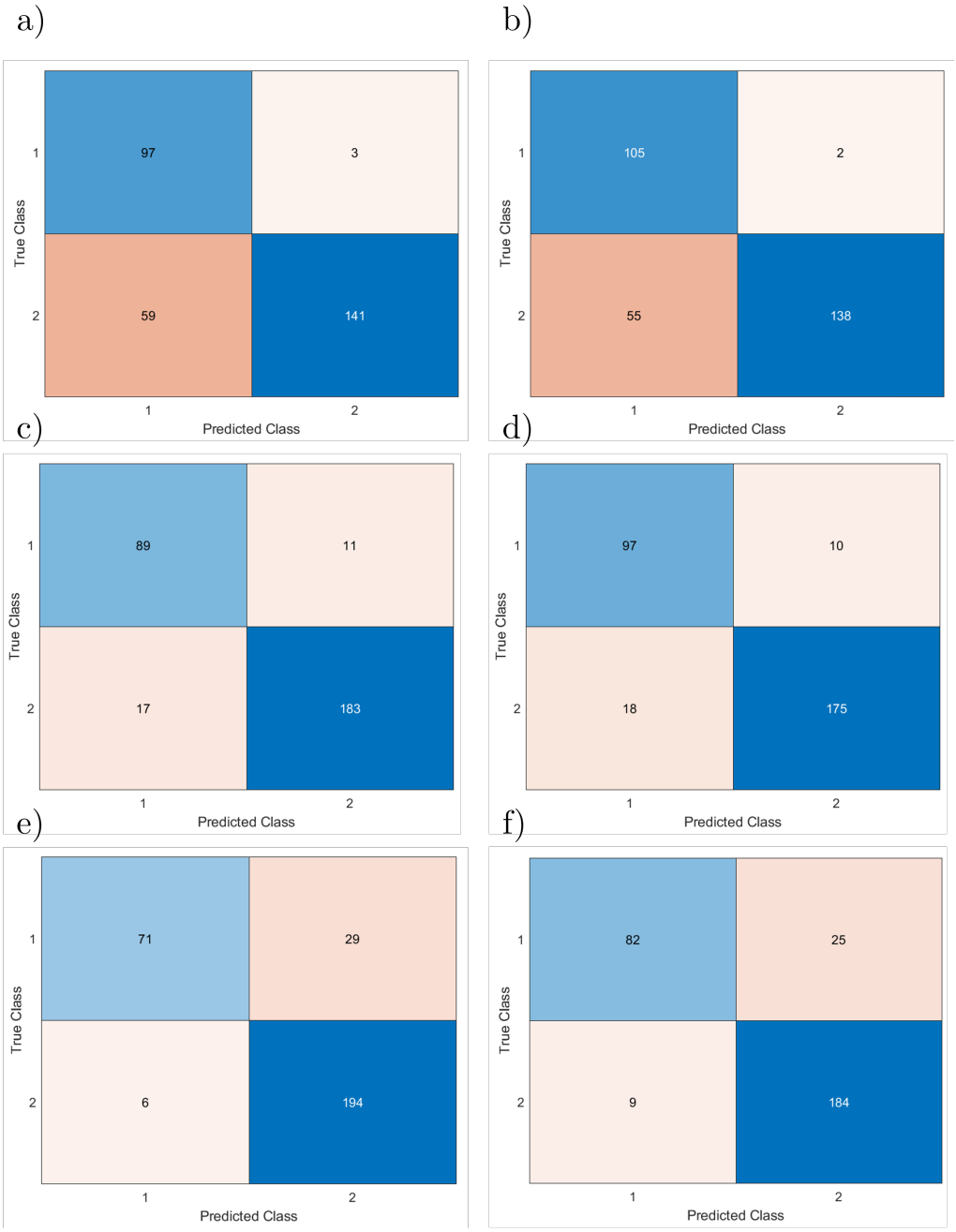

Figure 6: *Confusion matrix for the* ***bayesian*** *model with some prior probabilities values. In a), c) and e) the training data while in b), d) and f) the validation data. In a) and b) the initial guess is* $[1\ \ 0.1]$*, in c) and d) is* $[1\ \ 2]$*, while in e) and f) is* $[1\ \ 10]$*.*

The respective precision values are as follows:

$$\begin{aligned} \mathrm{ACC}_a &= 79.33\,\%\,; \quad \mathrm{ACC}_b = 81\,\% \\ \mathrm{ACC}_c &= 90.67\,\%\,; \quad \mathrm{ACC}_d = 90.67\,\% \\ \mathrm{ACC}_e &= 88.33\,\%\,; \quad \mathrm{ACC}_f = 88.67\,\% \end{aligned}$$

1.3 Relevant Code

The code prepared for this assignment is shown in the next pages in a wider style.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Master in Robotics
%                       Applied Artificial Intelligence
%
% Assinment 3.1:  k-nn and Bayesian Classifier
% Student: Josep Barbera Civera
% ID: 17048
% Date: 11/03/2024
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 0. Load data_D2_C2: p.value and t.value
% 1. Use the nn-classifier for computing the number of
%   missclassifications errors on p.value and on t.value,
%   using in both cases p.value as the learning data.
% 2. Same as before using the Bayesian classifier
% 3. Compute the miss-classification error on t.value when
%   using a Bayessian classifier with following a priori
%   probabilities: [1 0.1], [1 2] & [1 10]. Discuss the results.

load data_D2_C2.mat

%% Accesing Data
pvalues = p.value;
plabels = p.class;
tvalues = t.value;
tlabels = t.class;

[Mp, Np] = size(pvalues);
[Mt, Nt] = size(tvalues);

%% Models Trainning
% knn = fitcknn(pvalues', plabels');
% bayes = fitcnb(pvalues', plabels');
% prior = [1 0.1];
% prior = [1 2];
prior = [1 10];
bayes= fitcnb(pvalues', plabels','Prior', prior);

%% Model Prediction: training set
for i=1:Np
    % predKnn(:,i) = predict(knn, pvalues(:,i)');
    predBayes(:,i) = predict(bayes, pvalues(:,i)');
end
pred = predBayes;
%% Confusion matrix plotting
C = confusionmat(plabels',pred');
confusionchart(C);
set(gcf, 'Color', 'white');
saveas(gcf, 'confusion_chart_pvalues_bayes_prior_3.png'); % do not
   forget to change the name!

%% Model Prediction: test set
for i=1:Nt
```

```
    % predKnn(:,i) = predict(knn, tvalues(:,i)');
    predBayes(:,i) = predict(bayes, tvalues(:,i)');
end
pred = predBayes;
%% Confusion matrix plotting
C = confusionmat(tlabels',pred');
confusionchart(C);
set(gcf, 'Color', 'white');
saveas(gcf, 'confusion_chart_tvalues_bayes_prior_3.png'); % do not forget to change the name!

%% pvalues prediction map vs predicted values
% model = knn;
model = bayes;
x1 = min(pvalues(1,:)):0.01:max(pvalues(1,:));
x2 = min(pvalues(2,:)):0.01:max(pvalues(2,:));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)];
allPred = predict(model,XGrid);

figure
gscatter(XGrid(:,1),XGrid(:,2),allPred,[0.8500 0.3250 0.0980;0 0.4470 0.7410])
hold on

X = pvalues';
y = plabels';

h1 = plot(X(y == 1, 1), X(y == 1, 2), '.', 'MarkerSize', 10, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', "w");
hold on;
h2 = plot(X(y == 2, 1), X(y == 2, 2), '*', 'MarkerSize', 5, 'MarkerEdgeColor', 'w', 'MarkerFaceColor', "w");

xlabel('x')
ylabel('y')
title('{\bf knn classifier - tvalues (validation set)}')
legend([h1, h2], {'Label 1', 'Label 2'}, 'Location', 'best');
legend_box = findobj(gcf, 'Type', 'legend');
set(legend_box, 'Color', [0.8 0.8 0.8]); % Gray background color
axis tight
hold off
set(gcf, 'Color', 'white');
saveas(gcf, 'prediction_map_pvalues_bayes_prior_3.png'); % do not forget to change the name!


%% tvalues prediction map vs predicted values
% model = knn;
model = bayes;
x1 = min(tvalues(1,:)):0.01:max(tvalues(1,:));
x2 = min(tvalues(2,:)):0.01:max(tvalues(2,:));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)];
```

```
allPred = predict(model,XGrid);

figure
gscatter(XGrid(:,1),XGrid(:,2),allPred,[0.8500 0.3250 0.0980;0 0.4470
   0.7410])
hold on

X = tvalues';
y = tlabels';

h1 = plot(X(y == 1, 1), X(y == 1, 2), '.', 'MarkerSize', 10, '
   MarkerEdgeColor', 'k', 'MarkerFaceColor', "w");
hold on;
h2 = plot(X(y == 2, 1), X(y == 2, 2), '*', 'MarkerSize', 5, '
   MarkerEdgeColor', 'w', 'MarkerFaceColor', "w");

xlabel('x')
ylabel('y')
title('{\bf knn classifier - tvalues (validation set)}')
legend([h1, h2], {'Label 1', 'Label 2'}, 'Location', 'best');
legend_box = findobj(gcf, 'Type', 'legend');
set(legend_box, 'Color', [0.8 0.8 0.8]); % Gray background color
axis tight
hold off
set(gcf, 'Color', 'white');
saveas(gcf, 'prediction_map_tvalues_bayes_prior_3.png'); % do not forget
    to change the name!
```