```
clear all
rng("default")

%load the test data
TestData = csvread('.\Test1.csv',1);
%load mat file
mat = load('.\Final_RF.mat');

%call model from mat file
RFmodel_best = mat.RFmodel_best;


%feature selection random grid search
TestData =TestData(:,[1,2,3,4,5,10]);

% Hyperparametes list
n_estimators = [5,10,20,40,80,160,300];
n_predictors = [2,3,4];

% Initialize the hp_perf matrix with the correct size
%this matrix contains the performance of different combination of
%hyperparameters
hp_perf = mat.hp_perf;
cv_results = mat.cv_results;
training_times = mat.training_times;

% Find the maximum value in the hp_perf matrix
[max_value, max_idx] = max(hp_perf(:));

% Find the indices of the maximum value in the hp_perf matrix
[i, j] = ind2sub(size(hp_perf), max_idx);

% Get the values of the hyperparameters that gave the best performance
best_n_estimators = n_estimators(i);
best_n_predictors = n_predictors(j);

% Compute the minimum, maximum, and average training time
min_training_time = min(training_times(:));
max_training_time = max(training_times(:));
mean_training_time = mean(training_times(:));


% Evaluate the model on the test set
X_test = TestData(:,1:end-1);
y_test = TestData(:,end);

tic
predictions = predict(RFmodel_best,X_test);
predict_time = toc;
predictions = str2num(cell2mat(predictions));
iscorrect = predictions == y_test;
```

```
test_accuracy = sum(iscorrect) / numel(predictions);


% Compute the confusion matrix
cm = confusionmat(y_test, predictions);

hold off
%plot confusion matrix
confusionchart(y_test,predictions,"Normalization","absolute")
confusionchart(y_test,predictions,"Normalization","row-normalized")
confusionchart(y_test,predictions,"Normalization","column-normalized")

%copute percision recall and f1 score
precision = cm(2,2) / (cm(2,2) + cm(1,2));
recall = cm(2,2) / (cm(2,2) + cm(2,1));
f1 = 2 * precision * recall / (precision + recall);

% Get the predicted probabilities for the test set
[predictions, scores] = predict(RFmodel_best,X_test);
% Convert the predicted labels to a binary vector
predictions = str2num(cell2mat(predictions));
% Compute the ROC curve
[fpr, tpr, thr] = perfcurve(y_test, scores(:,2), 1);
% Compute the AUC value
auc = trapz(fpr,tpr);
% Plot the ROC curve
figure;
plot(fpr,tpr);
xlabel('False Positive Rate');
ylabel('True Positive Rate');
title(sprintf('ROC curve (AUC = %0.2f)', auc));


%plot avg accuracy vs number of trees
accuracy_matrix = zeros(length(n_estimators));
% Create a new figure
figure;

% Loop through each row of the hp_perf matrix
for i = 1:length(n_estimators)
    % Extract the average accuracy for each value of n_estimators
    avg_accuracy = mean(hp_perf(i,:));

    accuracy_matrix(i) = avg_accuracy;
end
% Plot the average accuracy as a function of n_estimators
plot(n_estimators, accuracy_matrix);

% Set the y-axis limits to be from 0.65 to 0.80
ylim([0.6 0.75])
```

```
% Add a title and axis labels to the plot
title('Average Accuracy vs. Num Trees');
xlabel('Num Trees');
ylabel('Average Accuracy');



%plot average accuracy vs number of predictors
accuracy_matrix = zeros(length(n_predictors));
% Create a new figure
figure;

% Loop through each row of the hp_perf matrix
for i = 1:length(n_predictors)
    % Extract the average accuracy for each value of n_estimators
    avg_accuracy = mean(hp_perf(:,i));

    accuracy_matrix(i) = avg_accuracy;
end
% Plot the average accuracy as a function of n_estimators
plot(n_predictors, accuracy_matrix);

% Set the y-axis limits to be from 0.65 to 0.80
ylim([0.65 0.75])
xticks(n_predictors);

% Add a title and axis labels to the plot
title('Average Accuracy vs. Num Predictors');
xlabel('Num predectors');
ylabel('Average Accuracy');



%plot average training time vs number of predictors
train_matrix = zeros(length(n_predictors));
% Create a new figure
figure;

% Loop through each row of the hp_perf matrix
for i = 1:length(n_predictors)
    % Extract the average accuracy for each value of n_estimators
    avg_traintime = mean(training_times(:,i));

    train_matrix(i) = avg_traintime;
end
% Plot the average accuracy as a function of n_estimators
plot(n_predictors, train_matrix);

xticks(n_predictors);
```

```
% Add a title and axis labels to the plot
title('Average training time vs. Num Predictors');
xlabel('Num of Predictors');
ylabel('Average training time');



%plot average training time vs number of trees
train_matrix = zeros(length(n_estimators));
% Create a new figure
figure;

% Loop through each row of the hp_perf matrix
for i = 1:length(n_estimators)
    % Extract the average accuracy for each value of n_estimators
    avg_traintime = mean(training_times(i,:));

    train_matrix(i) = avg_traintime;
end
% Plot the average accuracy as a function of n_estimators
plot(n_estimators, train_matrix);

% Add a title and axis labels to the plot
title('Average training time vs. Num Trees');
xlabel('Num trees');
ylabel('Average training time');
```