Written by# Saroj Shah

Date# 04-13-2025 Referance: Elgala, Hany. Class Notes, “Introduction to Machine Learning for Engineer IECE 565.” Hahn, Brian D., and Daniel T. Valentine. Essential MATLAB for Engineers and Scientists. Amsterdam, Academic Press, 2019. Harley, C., R. Reynolds, and M. Noordewier. "Molecular Biology (Promoter Gene Sequences)." UCI Machine Learning Repository, 1987, https://doi.org/10.24432/C5S01D. Accessed 4 Feb. 2025. ajitsingh98. “GitHub - Ajitsingh98/DNA-Classification-Machine-Learning-Project: Classify DNA Sequence into Binary Class Using Different Classification Algorithms.” GitHub, 2019, github.com/ajitsingh98/DNA-Classification-Machine-Learning-Project. Accessed 4 Feb. 2025. ChatGPT

DNA Classification Using Machine Learning

About: In this project, we will explore the world of bioinformatics by using Markov models, K-nearest neighbor (KNN) algorithms, support vector machines, and other common classifiers to classify short E. Coli DNA sequences. This project will use a dataset from the UCI Machine Learning Repository that has 105 DNA sequences, with 57 sequential nucleotides (“base-pairs”) each.

It includes: - Importing data from the UCI repository - Converting text inputs to numerical data - Building and training classification algorithms - Comparing and contrasting classification algorithms

Cleaning environment

```
clc; clear; close all;

% Hide warnings
warning('off', 'all');  % Turn off all warnings
```

Step 1: Importing the Dataset

```
% In this step, we will import the necessary data from the UCI repository or from
store drive
% The data will be read into MATLAB and processed as a table.

% % Dowloading the file from online URL location
% url ='https://archive.ics.uci.edu/ml/machine-learning-databases/molecular-biology/
promoter-gene-sequences/promoters.data';
% filename = 'promoters.data';
%
% if exist(filename, 'file'), delete(filename); end
% websave(filename, url);

% Downloading file from saved location (Specifying the path to the file)
file_path = 'C:\Users\Saroj\Documents\MATLAB\ML Project\promoters.data';
filename = 'promoters.data';

% Checking if the file exists, and delete it if necessary
if exist(filename, 'file')
    delete(filename);  % Delete the existing file
```

```
end
filename = file_path;

% Import the dataset with correct FileType and delimiter
data = readtable(filename, 'FileType', 'text', 'Delimiter', ',', 'Format',
'%s%s%s'); % 'Class', 'id', 'Sequence'

% Get the size of the data table
size_of_data = size(data);
fprintf('The Size of data table is: %d rows, %d columns\n', size_of_data);
```

```
The Size of data table is: 105 rows, 3 columns
```

```

% Rename the columns
data.Properties.VariableNames = {'Class', 'id', 'Sequence'};

% Convert 'Class' and 'id' columns from cell arrays to string arrays
data.Class = string(data.Class);
data.id = string(data.id);

% Convert 'Sequence' column to a string array
data.Sequence = string(data.Sequence);

% Display first 5 rows to verify
disp('First 5 rows of the dataset to verify:');
```

```
First 5 rows of the dataset to verify:
```

```
disp(data(1:5, :));
```

```
    Class        id                                  Sequence                            
    _____    ____________    ___________________________________________________________

     "+"     "AMPC"          "tgctatcctgacagttgtcacgctgattggtgtcgttacaatctaacgcatcgccaa"
     "+"     "AROH"          "gtactagagaactagtgcattagcttatttttttgttatcatgctaaccacccggcg"
     "+"     "DEOP2"         "aattgtgatgtgtatcgaagtgtgttgcggagtagatgttagaatactaacaaactc"
     "+"     "LEU1_TRNA"     "tcgataattaactattgacgaaaagctgaaaaccactagaatgcgcctccgtggtag"
     "+"     "MALEFG"        "aggggcaaggaggatggaaagaggttgccgtataaagaaactagagtccgtttaggt"
```

```

% Accessing the 'Sequence' and 'Class' columns
sequences = data.Sequence;  % Sequences (DNA sequences)
labels = data.Class;        % Labels (class labels)
```

Step 2: Preprocessing the Dataset

```
% Clean sequences by removing tabs and non-alphabet characters
for i = 1:length(sequences)
    sequence = sequences{i};                                           % Accessing
sequence
    cleaned_sequence = regexprep(sequence, '\t', '');                  % Removing
tabs
```

```
    cleaned_sequence = regexprep(cleaned_sequence, '[^acgtACGT]', '');  % Keeping
only valid characters (A, C, G, T)
    sequences{i} = lower(cleaned_sequence);                             % Update
with cleaned sequence (converted to lowercase)
end
```

Step 3: Encoding Sequences

```
numSequences = length(sequences);
encodedSequences = zeros(numSequences, 57);                     % Since nucleotide
are 57 sequences long

for i = 1:numSequences
    seq = sequences{i};
    for j = 1:min(length(seq), 57)                              % Ensure it willn't
exceed sequence length
        switch upper(seq(j))
            case 'A', encodedSequences(i, j) = 1;
            case 'C', encodedSequences(i, j) = 2;
            case 'G', encodedSequences(i, j) = 3;
            case 'T', encodedSequences(i, j) = 4;
        end
    end
end
```

Step 4: Stratified Shuffle and Train-Test Split

```
%cv = cvpartition(labels, 'HoldOut', 0.3); % 30% test data      % 30% test data
without stratified split
cv = cvpartition(labels, 'HoldOut', 0.3, 'Stratify', true);     % 30% test data
with stratified split
X_train = encodedSequences(training(cv), :);
X_test = encodedSequences(test(cv), :);
y_train = labels(training(cv));
y_test = labels(test(cv));

fprintf('Training Data Size: %d\n', size(X_train, 1));
```

```
Training Data Size: 74
```

```
fprintf('Test Data Size: %d\n', size(X_test, 1));
```

```
Test Data Size: 31
```

Step 5: Checking and Removeing Constant Features

```
% Computing variance for each feature in X_train
feature_variances = var(X_train, 0, 1);                     % Computeing variance
for each feature
```

```
constant_features = find(feature_variances == 0);           % Finding indices of
features with zero variance

% Displaying constant features if it is found
if ~isempty(constant_features)
    disp('Constant features found at indices:');
    disp(constant_features);
end

% Removing constant features from the dataset (both training and testing data)
X_train_cleaned = X_train(:, feature_variances > 0);
X_test_cleaned = X_test(:, feature_variances > 0);

% Converting labels to numeric (1 for '+' and 0 for '-')
y_test = strcmp(y_test, '+');                               % Convert '+' to 1, '-'
to 0 (true/false)
y_train = strcmp(y_train, '+');                             % Same conversion for
training labels
```

Step 6: Training Classifiers

```
% Decision Tree classifier with max depth control using 'MaxNumSplits'
dtModel = fitctree(X_train_cleaned, y_train, 'MaxNumSplits', 5);              %
MaxNumSplits limits depth by number of splits

% Other Classifiers
svmModel = fitcsvm(X_train_cleaned, y_train, 'KernelFunction', 'linear');       %
Linear SVM (support vector machines)
knnModel = fitcknn(X_train_cleaned, y_train, 'NumNeighbors', 3);              %
k-Nearest Neighbors (k-NN)
nbModel = fitcnb(X_train_cleaned, y_train);                                   %
Naive Bayes classifier = probalistic classifier based on Bayes' thm.
rfModel = fitcensemble(X_train_cleaned, y_train, 'Method', 'Bag', ...
    'NumLearningCycles', 10, 'Learners', 'tree');                             %
Random Forest
mlpModel = fitcnet(X_train_cleaned, y_train, 'Lambda', 1);                    %
Neural Network
```

Step 7: Evaluating Models

```
y_pred_svm = predict(svmModel, X_test_cleaned);  % Predictions should already be
numeric
y_pred_knn = predict(knnModel, X_test_cleaned);  % Predictions should already be
numeric
y_pred_nb = predict(nbModel, X_test_cleaned);  % Predictions should already be
numeric
y_pred_dt = predict(dtModel, X_test_cleaned);  % Predictions from Decision Tree
y_pred_rf = predict(rfModel, X_test_cleaned);  % Predictions from Random Forest
y_pred_mlp = predict(mlpModel, X_test_cleaned);  % Predictions from Neural Network
```

```
% Check if predictions and y_test are correct
disp('First 5 values of y_test:');
```

```
First 5 values of y_test:
```

```
disp(y_test(1:5));  % Display the first 5 values of y_test
```

```
     1
     1
     1
     1
     1
```

```


disp('First 5 values of y_pred_svm:');
```

```
First 5 values of y_pred_svm:
```

```
disp(y_pred_svm(1:5));  % Display the first 5 predictions from SVM
```

```
     0
     0
     1
     1
     1
```

Step 8: Computing accuracy for each classifier

```
acc_svm = sum(y_pred_svm == y_test) / length(y_test);
acc_knn = sum(y_pred_knn == y_test) / length(y_test);
acc_nb = sum(y_pred_nb == y_test) / length(y_test);
acc_dt = sum(y_pred_dt == y_test) / length(y_test);
acc_rf = sum(y_pred_rf == y_test) / length(y_test);
acc_mlp = sum(y_pred_mlp == y_test) / length(y_test);

fprintf('SVM Accuracy: %.2f%%\n', acc_svm * 100);
```

```
SVM Accuracy: 74.19%
```

```
fprintf('k-NN Accuracy: %.2f%%\n', acc_knn * 100);
```

```
k-NN Accuracy: 74.19%
```

```
fprintf('Naïve Bayes Accuracy: %.2f%%\n', acc_nb * 100);
```

```
Naïve Bayes Accuracy: 87.10%
```

```
fprintf('Decision Tree Accuracy: %.2f%%\n', acc_dt * 100);
```

```
Decision Tree Accuracy: 74.19%
```

```
fprintf('Random Forest Accuracy: %.2f%%\n', acc_rf * 100);
```

```
Random Forest Accuracy: 80.65%
```

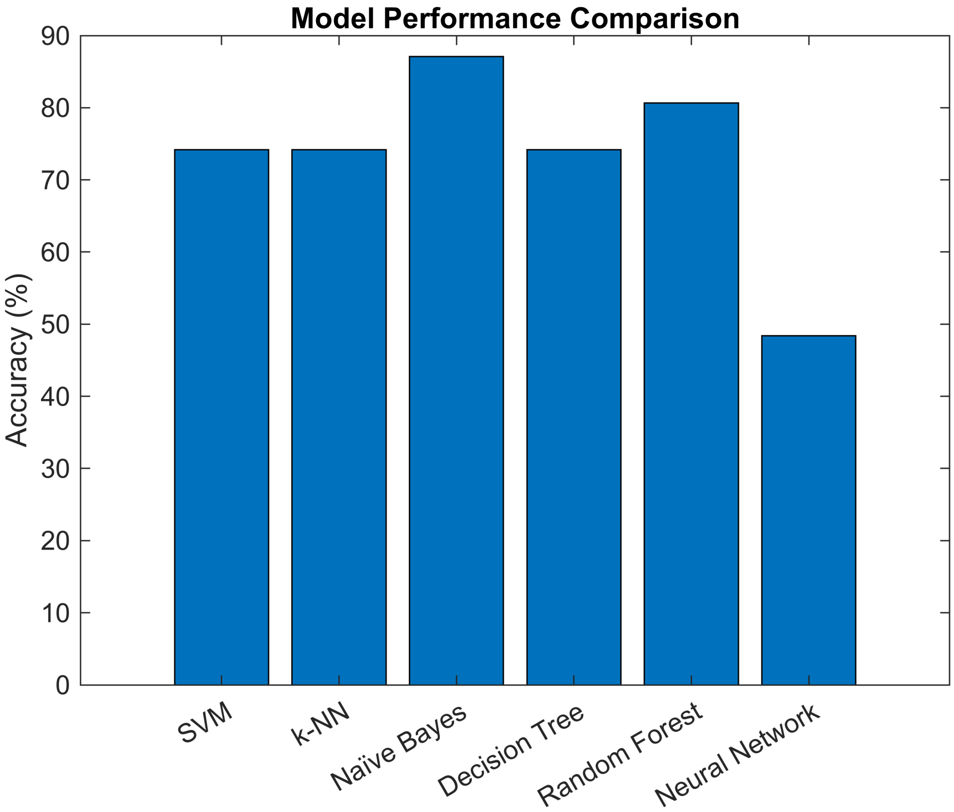
```
fprintf('Neural Network Accuracy: %.2f%%\n', acc_mlp * 100);
```

```
Neural Network Accuracy: 48.39%
```

Step 9: Visualizing Results

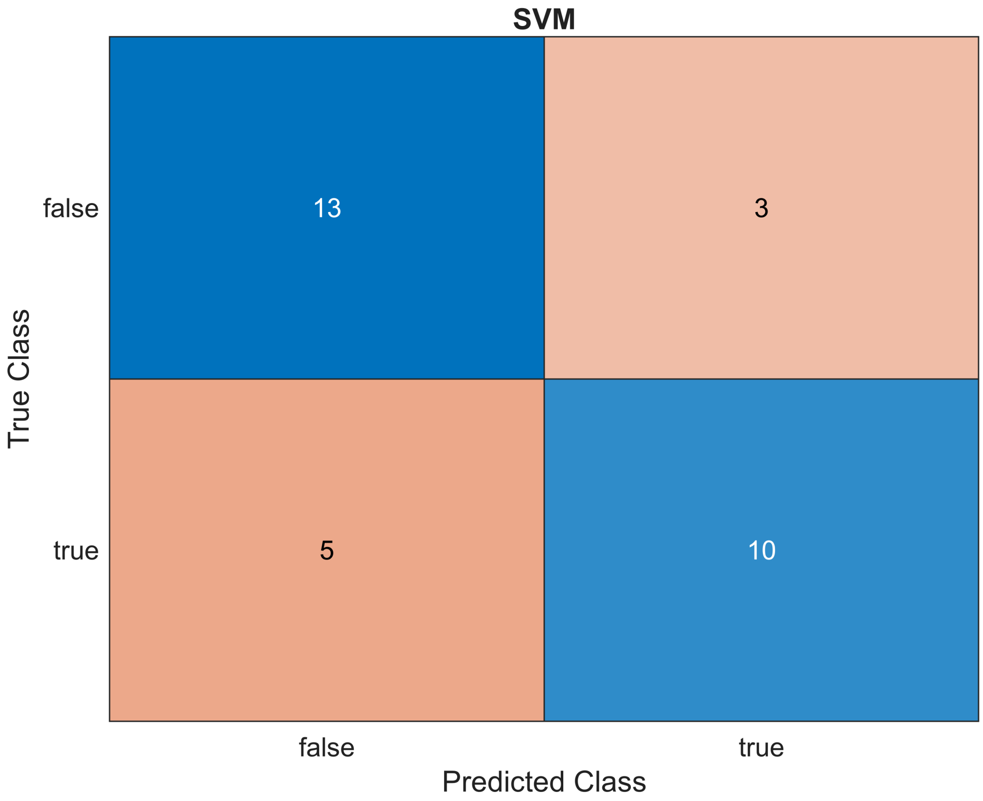
Compare classifier performances using a bar chart

```
figure;
bar([acc_svm, acc_knn, acc_nb, acc_dt, acc_rf, acc_mlp] * 100);
set(gca, 'xticklabel', {'SVM', 'k-NN', 'Naïve Bayes', 'Decision Tree', 'Random
Forest', 'Neural Network'});
ylabel('Accuracy (%)');
title('Model Performance Comparison');
```

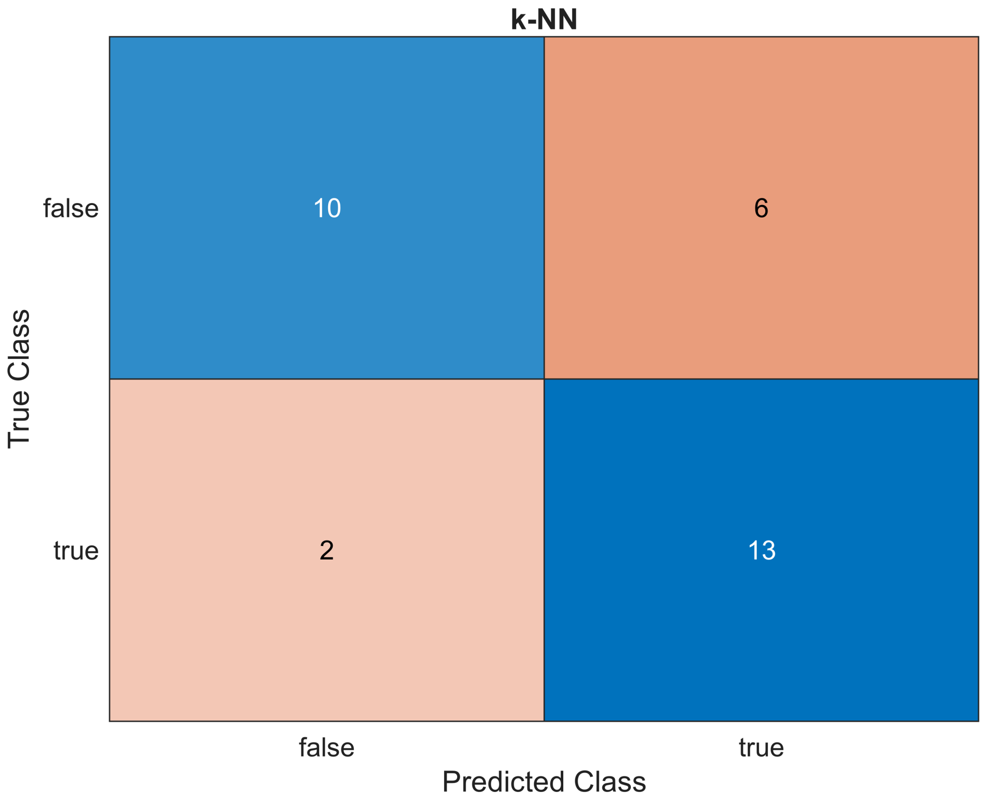


Step 10: Plotting Confusion Matrix for each classifier

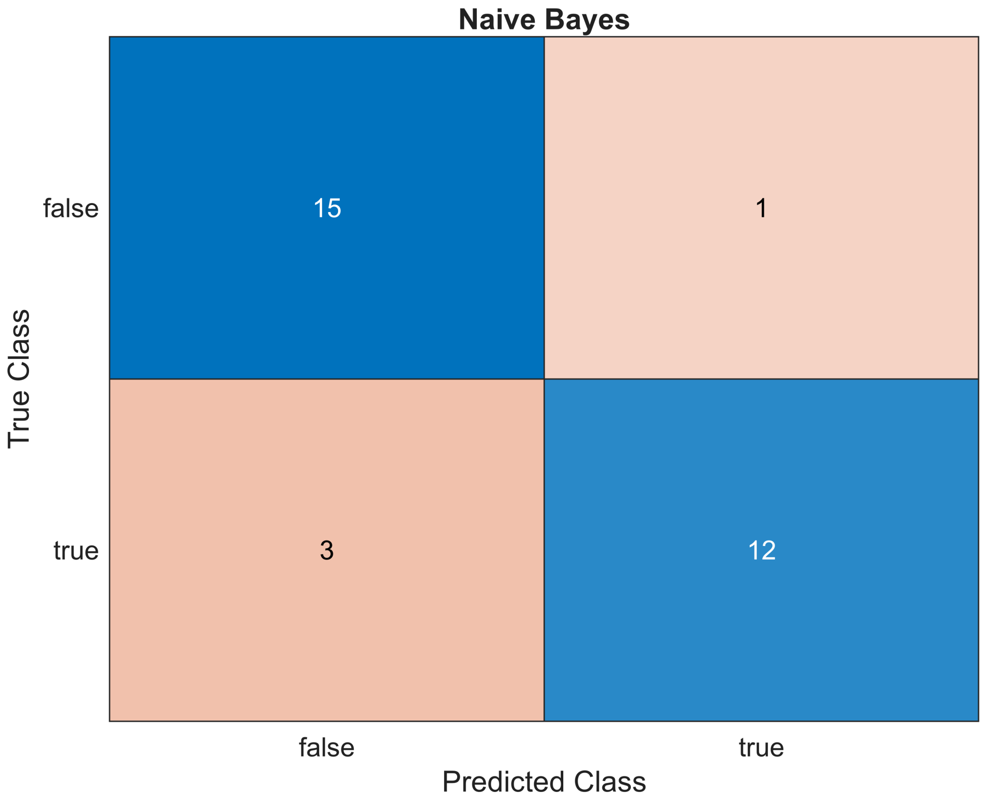
```
figure;
confusionchart(y_test, y_pred_svm, 'Title', 'SVM');
```



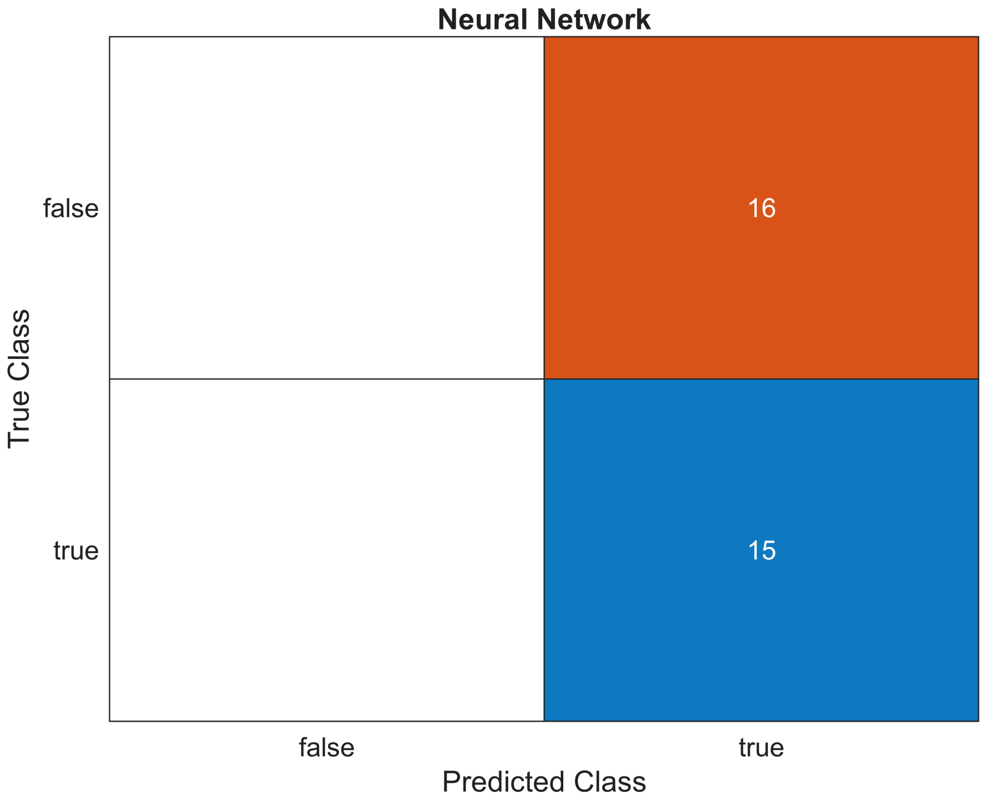
```
figure;
confusionchart(y_test, y_pred_knn, 'Title', 'k-NN');
```



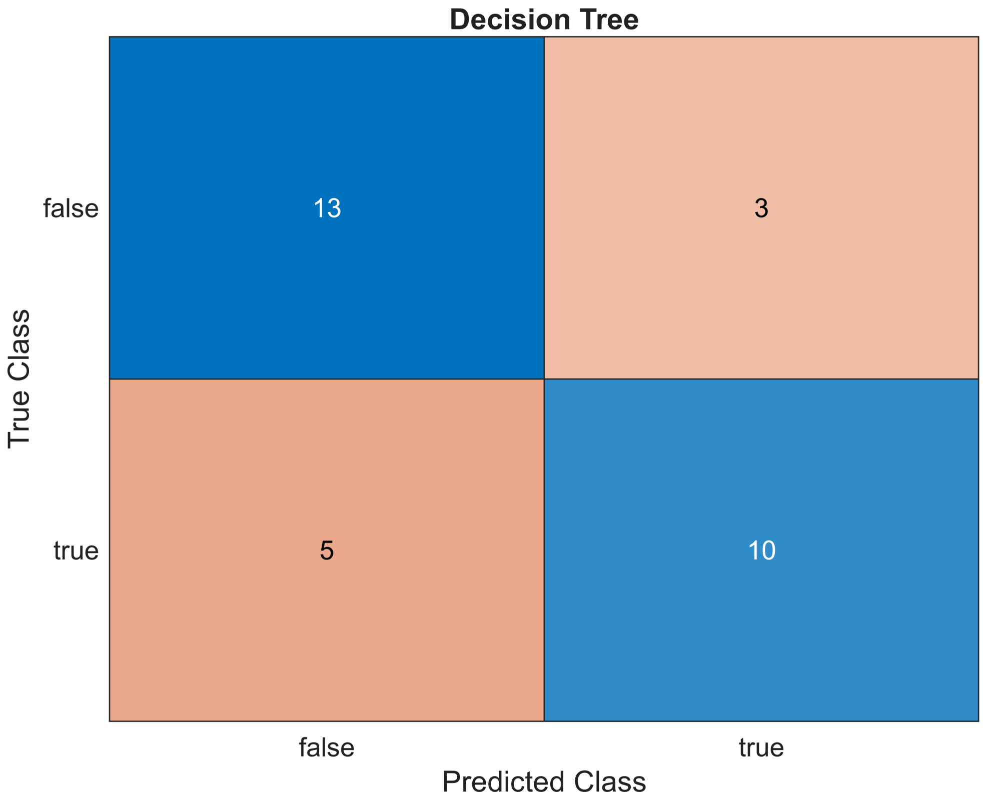
```
figure;
confusionchart(y_test, y_pred_nb, 'Title', 'Naive Bayes');
```



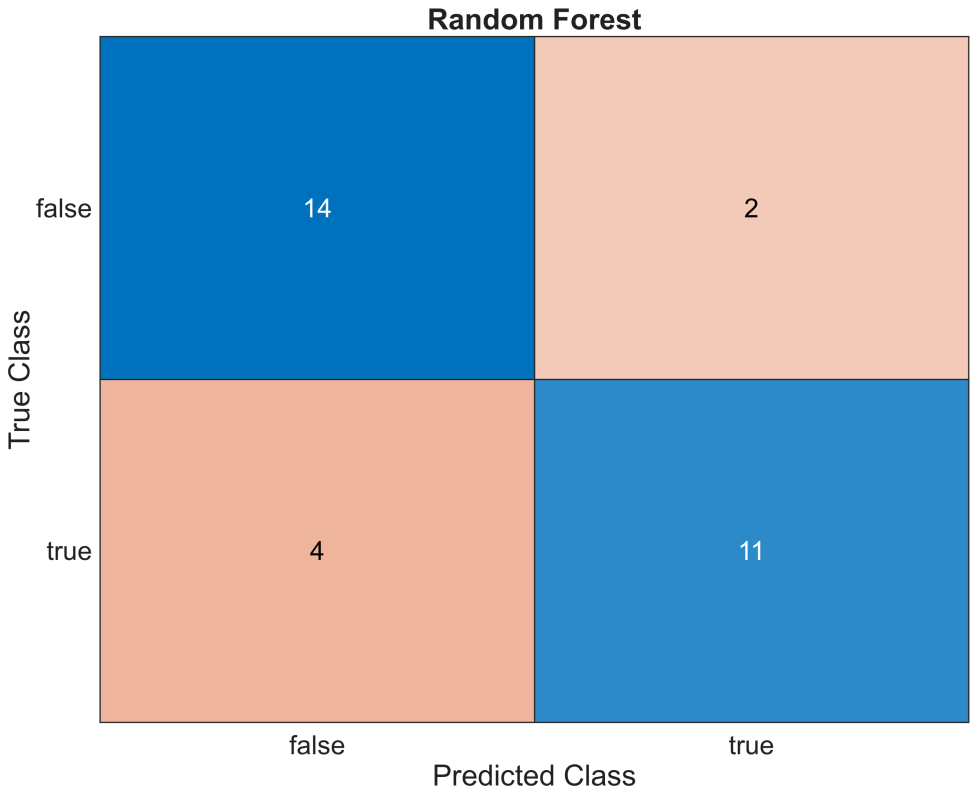
```
figure;
confusionchart(y_test, y_pred_mlp, 'Title', 'Neural Network');
```



```
figure;
confusionchart(y_test, y_pred_dt, 'Title', 'Decision Tree');
```



```
figure;
confusionchart(y_test, y_pred_rf, 'Title', 'Random Forest');
```



Step 11: The End

```
fprintf('\n\tThank you!\n\tSaroj Shah\n');
```

```
    Thank you!
    Saroj Shah
```