In [39]:

```
import sys
import numpy as np
import pandas as pd
import sklearn as skl
import matplotlib as mpl

print('Python: {}'.format(sys.version))
print('Numpy: {}'.format(np.__version__))
print('Pandas: {}'.format(pd.__version__))
print('Sklearn: {}'.format(skl.__version__))
print('Matplotlib: {}'.format(mpl.__version__))
```

```
Python: 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Numpy: 1.18.1
Pandas: 0.24.2
Sklearn: 0.22.1
Matplotlib: 3.1.3
```

In [40]:

```
# Import the dataset
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/molecular-biology/prom
oter-gene-sequences/promoters.data'
names = ['Class', 'id', 'Sequence']
df = pd.read_csv(url,names=names)
```

In [41]:

```
df
```

Out[41]:

	Class	id	Sequence
0	+	S10	\t\ttactagcaatacgcttgcgttcggtggttaagtatgtataat...
1	+	AMPC	\t\ttgctatcctgacagttgtcacgctgattggtgtcgttacaat...
2	+	AROH	\t\tgtactagagaactagtgcattagcttatttttttgttatcat...
3	+	DEOP2	\taattgtgatgtgtatcgaagtgtgttgcggagtagatgttagaa...
4	+	LEU1_TRNA	\ttcgataattaactattgacgaaaagctgaaaaccactagaatgc...
5	+	MALEFG	\taggggcaaggaggatggaaagaggttgccgtataaagaaactag...
6	+	MALK	\t\tcaggggtggaggatttaagccatctcctgatgacgcatagt...
7	+	RECA	\t\ttttctacaaaacacttgatactgtatgagcatacagtataat...
8	+	RPOB	\t\tcgacttaatatactgcgacaggacgtccgttctgtgtaaatc...
9	+	RRNAB_P1	\tttttaaatttcctcttgtcaggccggaataactccctataatgc...
10	+	RRNAB_P2	\tgcaaaaataaatgcttgactctgtagcgggaaggcgtattatgc...
11	+	RRNDEX_P2	\tcctgaaattcagggttgactctgaaagaggaaagcgtaatatac...
12	+	RRND_P1	\tgatcaaaaaaatacttgtgcaaaaaattgggatccctataatgc...
13	+	RRNE_P1	\tctgcaatttttctattgcggcctgcggagaactccctataatgc...
14	+	RRNG_P1	\ttttatatttttcgcttgtcaggccggaataactccctataatgc...
15	+	RRNG_P2	\taagcaaagaaatgcttgactctgtagcgggaaggcgtattatgc...
16	+	RRNX_P1	\tatgcatttttccgcttgtcttcctgagccgactccctataatgc...
17	+	TNAA	\t\taaacaatttcagaatagacaaaaactctgagtgtaataatgt...
18	+	TYRT	\t\ttctcaacgtaacactttacagcggcgcgtcatttgatatgat...
19	+	ARAC	\t\tgcaaataatcaatgtggacttttctgccgtgattatagacac...
20	+	LACI	\t\tgacaccatcgaatggcgcaaaacctttcgcggtatggcatga...
21	+	MALT	\t\taaaaacgtcatcgcttgcattagaaaggtttctggccgacct...
22	+	TRP	\t\ttctgaaatgagctgttgacaattaatcatcgaactagttaac...
23	+	TRPP2	\taccggaagaaaaccgtgacattttaacacgtttgttacaaggta...
24	+	THR	\t\taaattaaaattttattgacttaggtcactaaatactttaacc...
25	+	BIOB	\t\ttttgtcataatcgacttgtaaaccaaattgaaaagatttaggt...
26	+	FOL	\t\tcatcctcgcaccagtcgacgacggtttacgctttacgtatag...
27	+	UVRBP1	\ttccagtataatttgttggcataattaagtacgacgagtaaaatt...
28	+	UVRBP3	\tacagttatccactattcctgtggataaccatgtgtattagagtt...
29	+	LEXA	\t\ttgtgcagtttatggttccaaaatcgccttttgctgtatatac...
...	...	...	...
76	-	668	\t\tcatgtcagcctcgacaacttgcataaatgctttcttgtagac...
77	-	413	\t\taggaggaactacgcaaggttggaacatcggagagatgccagc...
78	-	991	\t\ttctcaacaagattaaccgacagattcaatctcgtggatggac...
79	-	751	\t\ttgaagtgcttagcttcaaggtcacggatacgaccgaagcgag...
80	-	850	\t\tctatatgcgctcatacgatatgaacgttgagactgccgctga...
81	-	93	\t\tgcggcagcacgtttccacgcggtgagagcctcaggattcatg...

	Class	id	Sequence
82	-	1108	\t\tatccctaatgtctacttccggtcaatccatctacgttaaccg...
83	-	915	\t\ttggcgtctatcggtgaacctccggtatcaacgctggaaggtg...
84	-	1019	\t\ttctcgtggatggacgttcaacattgaggaaggcataacgcta...
85	-	19	\t\ttattggcttgctcaagcatgaactcaaggctgatacggcgag...
86	-	1320	\t\ttagagggtgtactccaagaagaggaagatgaggctagacgtc...
87	-	91	\t\tcagcggcagcacgtttccacgcggtgagagcctcaggattca...
88	-	217	\t\ttttacgttggcgaccgctaggactttcttgttgattttccatg...
89	-	957	\t\tacgctaacgcagatgcagcgaacgctcggcgtattctcaaca...
90	-	260	\t\ttggtgttttgcgcaatgttaatcgctttgtacacctcaggcat...
91	-	557	\t\taaccattccggttgactcaatgagcatctcgatgcagcgtac...
92	-	1355	\t\tagacgtctctgcatggagtatgagatggactacggtgggtac...
93	-	244	\t\tttgttgattttccatgcggtgttttgcgcaatgttaatcgctt...
94	-	464	\t\tttgcacgggttgcgatagcctcagcgtattcaggtgcgagttc...
95	-	296	\t\ttaggcatgtaaacgtcttcgtagcgcatcagtgctttcttact...
96	-	648	\t\tccgagtagacccttagagagcatgtcagcctcgacaacttgc...
97	-	230	\t\tcgctaggactttcttgttgattttccatgcggtgttttgcgc...
98	-	1163	\t\tttatgaccgaacgagtcaatcagaccgctttgactctggtatt...
99	-	1321	\t\ttagagggtgtactccaagaagaggaagatgaggctagacgtct...
100	-	663	\t\ttgagagcatgtcagcctcgacaacttgcataaatgctttcttg...
101	-	799	\t\ttcctcaatggcctctaaacgggtcttgaggggtttttgctga...
102	-	987	\t\ttgtattctcaacaagattaaccgacagattcaatctcgtggat...
103	-	1226	\t\ttcgcgactacgatgagatgcctgagtgcttccgttactggatt...
104	-	794	\t\ttctcgtcctcaatggcctctaaacgggtcttgaggggtttttt...
105	-	1442	\t\tttaacattaataaataaggaggctctaatggcactcattagcc...

106 rows × 3 columns

In [42]:

```
df.describe()
```

Out[42]:

	Class	id	Sequence
count	106	106	106
unique	2	106	106
top	-	THR	\t\tttgtgcagtttatggttccaaaatcgccttttgctgtatatac...
freq	53	1	1

In [43]:

```
print(df.iloc[0])
```

```
Class                                                       +
id                                                        S10
Sequence    \t\ttactagcaatacgcttgcgttcggtggttaagtatgtataat...
Name: 0, dtype: object
```

In [44]:

```
print(df['Sequence'].iloc[0])
```

```
                tactagcaatacgcttgcgttcggtggttaagtatgtataatgcgcgggcttgtcgt
```

In [45]:

```
# Preprocessing the dataset

classes=df.loc[:,'Class']
#print(classes)

# generate list of DNA sequences
sequences=df.loc[:,'Sequence']
#print(sequences)

dataset={}
i=0

# loop through sequences and split into individual nucleotides
for seq in sequences:

    # split into nucleotides, remove tab characters
    nucleotides=list(seq)
    nucleotides=[x for x in seq if x!='\t']

    # append class assignment
    nucleotides.append(classes[i])

    # add to dataset
    dataset[i]=(nucleotides)

    #increment i
    i+=1

#print(dataset)
```

In [46]:

```
df=pd.DataFrame(dataset)
df=df.transpose()

# for clarity, lets rename the last dataframe column to class
df.rename(columns = {57: 'Class'}, inplace = True)

df
```

Out[46]:

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	Class
0	t	a	c	t	a	g	c	a	a	t	...	g	c	t	t	g	t	c	g	t	+
1	t	g	c	t	a	t	c	c	t	g	...	c	a	t	c	g	c	c	a	a	+
2	g	t	a	c	t	a	g	a	g	a	...	c	a	c	c	c	g	g	c	g	+
3	a	a	t	t	g	t	g	a	t	g	...	a	a	c	a	a	a	c	t	c	+
4	t	c	g	a	t	a	a	t	t	a	...	c	c	g	t	g	g	t	a	g	+
5	a	g	g	g	g	c	a	a	g	g	...	c	g	t	t	t	a	g	g	t	+
6	c	a	g	g	g	g	g	t	g	g	...	a	t	c	a	t	g	a	a	t	+
7	t	t	t	c	t	a	c	a	a	a	...	a	a	c	a	g	a	a	c	a	+
8	c	g	a	c	t	t	a	a	t	a	...	a	a	a	t	g	g	t	t	t	+
9	t	t	t	t	a	a	a	t	t	t	...	c	c	a	c	t	g	a	c	a	+
10	g	c	a	a	a	a	a	t	a	a	...	c	c	c	g	c	g	c	c	g	+
11	c	c	t	g	a	a	a	t	t	c	...	c	c	t	c	g	c	g	a	c	+
12	g	a	t	c	a	a	a	a	a	a	...	c	c	g	t	t	g	a	g	a	+
13	c	t	g	c	a	a	t	t	t	t	...	c	c	a	t	c	g	a	c	a	+
14	t	t	t	a	t	a	t	t	t	t	...	c	c	a	c	t	g	a	c	a	+
15	a	a	g	c	a	a	a	g	a	a	...	c	g	c	c	g	c	g	c	c	+
16	a	t	g	c	a	t	t	t	t	t	...	c	c	a	t	c	g	a	c	a	+
17	a	a	a	c	a	a	t	t	t	c	...	g	t	g	t	c	t	t	g	c	+
18	t	c	t	c	a	a	c	g	t	a	...	c	g	c	t	t	c	c	c	g	+
19	g	c	a	a	a	t	a	a	t	c	...	t	a	c	g	c	g	t	t	t	+
20	g	a	c	a	c	c	a	t	c	g	...	c	c	g	g	a	a	g	a	g	+
21	a	a	a	a	a	c	g	t	c	a	...	c	a	t	t	a	a	t	t	a	+
22	t	c	t	g	a	a	a	t	g	a	...	g	c	a	a	g	t	t	c	a	+
23	a	c	c	g	g	a	a	g	a	a	...	c	g	a	c	g	c	c	g	c	+
24	a	a	a	t	t	a	a	a	a	t	...	g	g	c	a	t	a	g	c	g	+
25	t	t	g	t	c	a	t	a	a	t	...	g	t	c	t	a	c	a	c	c	+
26	c	a	t	c	c	t	c	g	c	a	...	c	a	a	t	t	t	t	t	t	+
27	t	c	c	a	g	t	a	t	a	a	...	a	c	c	t	g	c	c	c	g	+
28	a	c	a	g	t	t	a	t	c	c	...	a	a	c	a	c	g	a	g	g	+
29	t	g	t	g	c	a	g	t	t	t	...	c	a	t	a	a	c	t	g	t	+
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
76	c	a	t	g	t	c	a	g	c	c	...	t	a	c	g	c	g	c	t	t	-
77	a	g	g	a	g	g	a	a	c	t	...	a	c	c	t	g	c	a	c	g	-
78	t	c	t	c	a	a	c	a	a	g	...	c	a	t	t	g	a	g	g	a	-
79	t	g	a	a	g	t	g	c	t	t	...	c	c	t	c	a	a	t	g	g	-
80	c	t	a	t	a	t	g	c	g	c	...	a	g	c	t	g	t	g	a	a	-
81	g	c	g	g	c	a	g	c	a	c	...	t	c	t	t	c	c	g	g	t	-

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	Class
82	a	t	c	c	c	t	a	a	t	g	...	c	t	a	t	g	t	g	t	a	-
83	t	g	g	c	g	t	c	t	a	t	...	a	c	g	c	a	g	a	t	g	-
84	t	c	t	c	g	t	g	g	a	t	...	g	a	t	g	t	t	t	a	c	-
85	t	a	t	t	g	g	c	t	t	g	...	g	a	g	c	c	t	t	g	t	-
86	t	a	g	a	g	g	g	t	g	t	...	t	g	g	a	g	t	a	t	g	-
87	c	a	g	c	g	g	c	a	g	c	...	t	g	t	c	t	t	c	c	g	-
88	t	t	a	c	g	t	t	g	g	c	...	t	t	t	g	c	g	c	a	a	-
89	a	c	g	c	t	a	a	c	g	c	...	a	c	c	g	a	c	a	g	a	-
90	g	g	t	g	t	t	t	t	g	c	...	g	t	c	t	t	c	g	t	a	-
91	a	a	c	c	a	t	t	c	c	g	...	a	t	g	a	a	t	a	g	a	-
92	a	g	a	c	g	t	c	t	c	t	...	c	t	g	g	a	t	g	g	a	-
93	t	g	t	t	g	a	t	t	t	t	...	c	c	t	c	a	g	g	c	a	-
94	t	g	c	a	c	g	g	g	t	t	...	c	t	c	a	g	a	g	t	c	-
95	a	g	g	c	a	t	g	t	a	a	...	a	c	g	c	a	c	c	a	g	-
96	c	c	g	a	g	t	a	g	a	c	...	g	c	t	t	t	c	t	t	g	-
97	c	g	c	t	a	g	g	a	c	t	...	a	a	t	c	g	c	t	t	t	-
98	t	a	t	g	a	c	c	g	a	a	...	a	a	c	a	t	t	a	t	t	-
99	a	g	a	g	g	g	t	g	t	a	...	g	g	a	g	t	a	t	g	a	-
100	g	a	g	a	g	c	a	t	g	t	...	t	g	c	c	c	t	a	c	g	-
101	c	c	t	c	a	a	t	g	g	c	...	g	a	a	c	t	a	t	a	t	-
102	g	t	a	t	t	c	t	c	a	a	...	t	c	a	a	c	a	t	t	g	-
103	c	g	c	g	a	c	t	a	c	g	...	a	a	g	g	c	t	t	c	c	-
104	c	t	c	g	t	c	c	t	c	a	...	a	g	g	a	g	g	a	a	c	-
105	t	a	a	c	a	t	t	a	a	t	...	t	c	a	a	g	a	a	c	t	-

106 rows × 58 columns

In [47]:

```
df.describe()
```

Out[47]:

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	
count	106	106	106	106	106	106	106	106	106	106	...	106	106	106	106	106	1
unique	4	4	4	4	4	4	4	4	4	4	...	4	4	4	4	4	
top	t	a	a	c	a	a	a	a	a	a	...	c	c	c	t	t	
freq	38	34	30	30	36	42	38	34	33	36	...	36	42	31	33	35	

4 rows × 58 columns

In [48]:

```
# Record value counts for each sequence
series = []
for name in df.columns:
    series.append(df[name].value_counts())

info = pd.DataFrame(series)
details = info.transpose()
print(details)
```

```
     0     1     2     3     4     5     6     7     8     9  ...    48  \
t  38.0  26.0  27.0  26.0  22.0  24.0  30.0  32.0  32.0  28.0  ...  21.0
c  27.0  22.0  21.0  30.0  19.0  18.0  21.0  20.0  22.0  22.0  ...  36.0
a  26.0  34.0  30.0  22.0  36.0  42.0  38.0  34.0  33.0  36.0  ...  23.0
g  15.0  24.0  28.0  28.0  29.0  22.0  17.0  20.0  19.0  20.0  ...  26.0
-   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN
+   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN

     49    50    51    52    53    54    55    56  Class
t  22.0  23.0  33.0  35.0  30.0  23.0  29.0  34.0    NaN
c  42.0  31.0  32.0  21.0  32.0  29.0  29.0  17.0    NaN
a  24.0  28.0  27.0  25.0  22.0  26.0  24.0  27.0    NaN
g  18.0  24.0  14.0  25.0  22.0  28.0  24.0  28.0    NaN
-   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   53.0
+   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   53.0

[6 rows x 58 columns]
```

In [49]:

```
# We can't run machine learning algorithms on the data in 'String' formats. We need to
 switch
# it to numerical data.
numerical_df = pd.get_dummies(df)

# We don't need both class columns.  Lets drop one then rename the other to simply 'Cla
ss'.
df = numerical_df.drop(columns=['Class_-'])

df.rename(columns = {'Class_+': 'Class'}, inplace = True)

print(df.iloc[:5])
```

```
   0_a  0_c  0_g  0_t  1_a  1_c  1_g  1_t  2_a  2_c  ...  54_t  55_a  55_c  
\
0    0    0    0    1    1    0    0    0    0    1  ...     0     0     0   
1    0    0    0    1    0    0    1    0    0    1  ...     0     1     0   
2    0    0    1    0    0    0    0    1    1    0  ...     0     0     1   
3    1    0    0    0    1    0    0    0    0    0  ...     0     0     0   
4    0    0    0    1    0    1    0    0    0    0  ...     1     1     0   

   55_g  55_t  56_a  56_c  56_g  56_t  Class  
0     1     0     0     0     0     1      1  
1     0     0     1     0     0     0      1  
2     0     0     0     0     1     0      1  
3     0     1     0     1     0     0      1  
4     0     0     0     0     1     0      1  

[5 rows x 229 columns]
```

In [66]:

```
# Use the model_selection module to separate training and testing datasets
from sklearn import model_selection
# Create X and Y datasets for training
X = np.array(df.drop(['Class'],1))
y = np.array(df['Class'])

# define seed for reproducibility
seed = 1

# split data into training and testing datasets
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2
0, random_state=seed)
```

In [113]:

```
# Now that we have our dataset, we can start building algorithms! We'll need to import
 each algorithm we plan on using
# from sklearn.  We also need to import some performance metrics, such as accuracy_scor
e and classification_report.

from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

# define scoring method
scoring = 'accuracy'

# Define models to train
models=[]

models.append(('Nearest Neighbors',KNeighborsClassifier(n_neighbors = 2)))
models.append(('Gaussian Process',GaussianProcessClassifier(1.0 * RBF(1.0))))
models.append(('Decision Tree',DecisionTreeClassifier(max_depth=5)))
models.append(('Random Forest',RandomForestClassifier(max_depth=5, n_estimators=10, max
_features=1)))
models.append(('Neural Net',MLPClassifier(alpha=1, max_iter=400, warm_start=True, verbo
se=0)))
models.append(('AdaBoost',AdaBoostClassifier()))
models.append(('Naive Bayes',GaussianNB()))
models.append(('SVM Linear',SVC(kernel = 'linear')))
models.append(('SVM RBF',SVC(kernel = 'rbf')))
models.append(('SVM Sigmoid',SVC(kernel = 'sigmoid')))
models.append(('SVM Polynomial',SVC(kernel = 'poly')))

# evaluate each model in turn
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, sco
ring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
Nearest Neighbors: 0.812500 (0.144771)
Gaussian Process: 0.868056 (0.104721)
Decision Tree: 0.758333 (0.146223)
Random Forest: 0.641667 (0.153131)
Neural Net: 0.905556 (0.073441)
AdaBoost: 0.906944 (0.068338)
Naive Bayes: 0.848611 (0.101198)
SVM Linear: 0.880556 (0.075103)
SVM RBF: 0.881944 (0.075116)
SVM Sigmoid: 0.891667 (0.066840)
SVM Polynomial: 0.881944 (0.075116)
```

In [114]:

```
from sklearn.metrics import classification_report, accuracy_score
for name, model in models:
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(name)
    print(accuracy_score(y_test, predictions))
    print(classification_report(y_test, predictions))
```

```
Nearest Neighbors
0.8636363636363636
              precision    recall  f1-score   support

           0       0.93      0.87      0.90        15
           1       0.75      0.86      0.80         7

    accuracy                           0.86        22
   macro avg       0.84      0.86      0.85        22
weighted avg       0.87      0.86      0.87        22

Gaussian Process
0.9545454545454546
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        15
           1       0.88      1.00      0.93         7

    accuracy                           0.95        22
   macro avg       0.94      0.97      0.95        22
weighted avg       0.96      0.95      0.96        22

Decision Tree
0.8636363636363636
              precision    recall  f1-score   support

           0       1.00      0.80      0.89        15
           1       0.70      1.00      0.82         7

    accuracy                           0.86        22
   macro avg       0.85      0.90      0.86        22
weighted avg       0.90      0.86      0.87        22

Random Forest
0.6818181818181818
              precision    recall  f1-score   support

           0       0.90      0.60      0.72        15
           1       0.50      0.86      0.63         7

    accuracy                           0.68        22
   macro avg       0.70      0.73      0.68        22
weighted avg       0.77      0.68      0.69        22

Neural Net
0.9545454545454546
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        15
           1       0.88      1.00      0.93         7

    accuracy                           0.95        22
   macro avg       0.94      0.97      0.95        22
weighted avg       0.96      0.95      0.96        22

AdaBoost
0.8181818181818182
              precision    recall  f1-score   support

           0       1.00      0.73      0.85        15
           1       0.64      1.00      0.78         7
    accuracy                           0.82        22
   macro avg       0.82      0.87      0.81        22
weighted avg       0.88      0.82      0.82        22

Naive Bayes
0.9090909090909091
              precision    recall  f1-score   support

           0       1.00      0.87      0.93        15
           1       0.78      1.00      0.88         7

    accuracy                           0.91        22
   macro avg       0.89      0.93      0.90        22
weighted avg       0.93      0.91      0.91        22

SVM Linear
0.9090909090909091
              precision    recall  f1-score   support

           0       1.00      0.87      0.93        15
           1       0.78      1.00      0.88         7

    accuracy                           0.91        22
   macro avg       0.89      0.93      0.90        22
weighted avg       0.93      0.91      0.91        22

SVM RBF
0.9090909090909091
              precision    recall  f1-score   support

           0       1.00      0.87      0.93        15
           1       0.78      1.00      0.88         7

    accuracy                           0.91        22
   macro avg       0.89      0.93      0.90        22
weighted avg       0.93      0.91      0.91        22

SVM Sigmoid
0.9090909090909091
              precision    recall  f1-score   support

           0       1.00      0.87      0.93        15
           1       0.78      1.00      0.88         7

    accuracy                           0.91        22
   macro avg       0.89      0.93      0.90        22
weighted avg       0.93      0.91      0.91        22

SVM Polynomial
0.9090909090909091
              precision    recall  f1-score   support

           0       1.00      0.87      0.93        15
           1       0.78      1.00      0.88         7

    accuracy                           0.91        22
   macro avg       0.89      0.93      0.90        22
weighted avg       0.93      0.91      0.91        22
```