Gustav Nilsson(guni1700), Elias Berglin(elbe1703) 2021-02-25

Laboration 2 - Python

Comparison between Naive Bayes and Decision Tree:

Decision Tree is unsupervised. Naive Bayes is supervised.

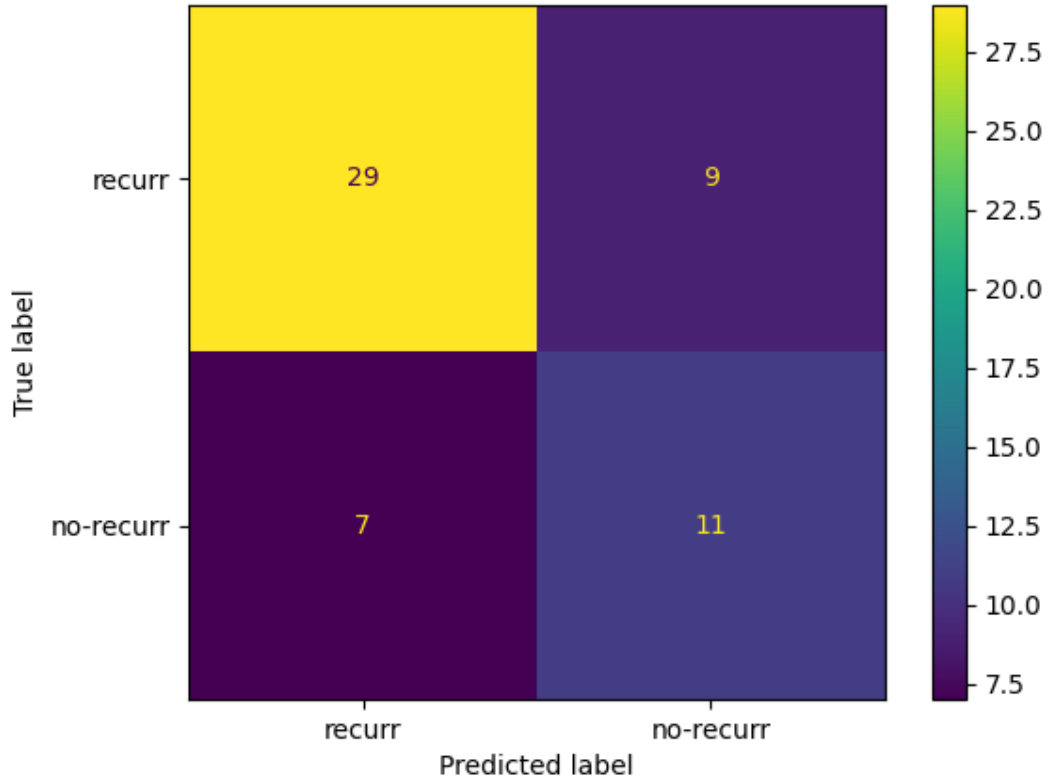
Is the tree reasonably pruned? What is the drawback of having too large or a too small tree?

Our tree is quite large and could be pruned a bit more we think.

The drawback of having too large decision tree is that it has a large chance of misclassifying and becoming too complex, since it has to go through so many nodes, while having a too small tree can make it too general, if it just splits half half it won't tell us much about the data it is classifying.

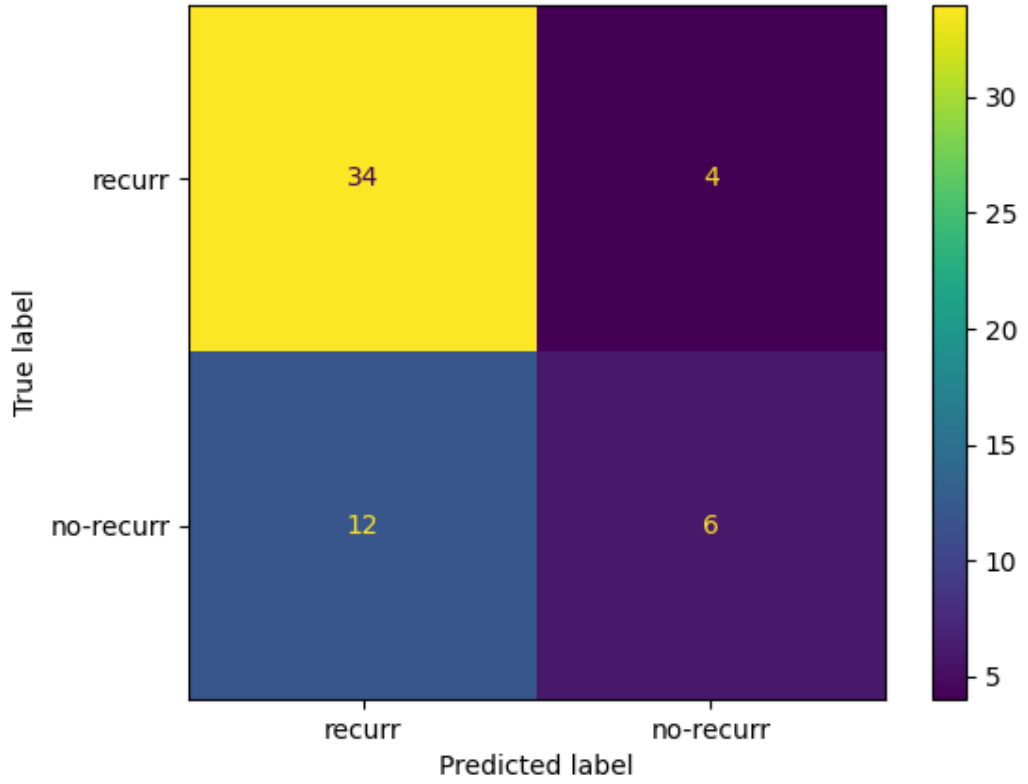
Determine which of the two methods is best in the case from a medical diagnostics point of view, in terms of true positive rate or true negative rate:

True positive rate Naive Bayes: 61.111%
True negative rate Naive Bayes: 76.316%



True/False Matrix	Predicted: Recurring	Predicted: no-recurring
Actual: Recurring	TN=29	FP=9
Actual: No-recurring	FN=7	TP=11

True positive rate Decision Tree: 33.333%
True negative rate Decision Tree: 81.579%



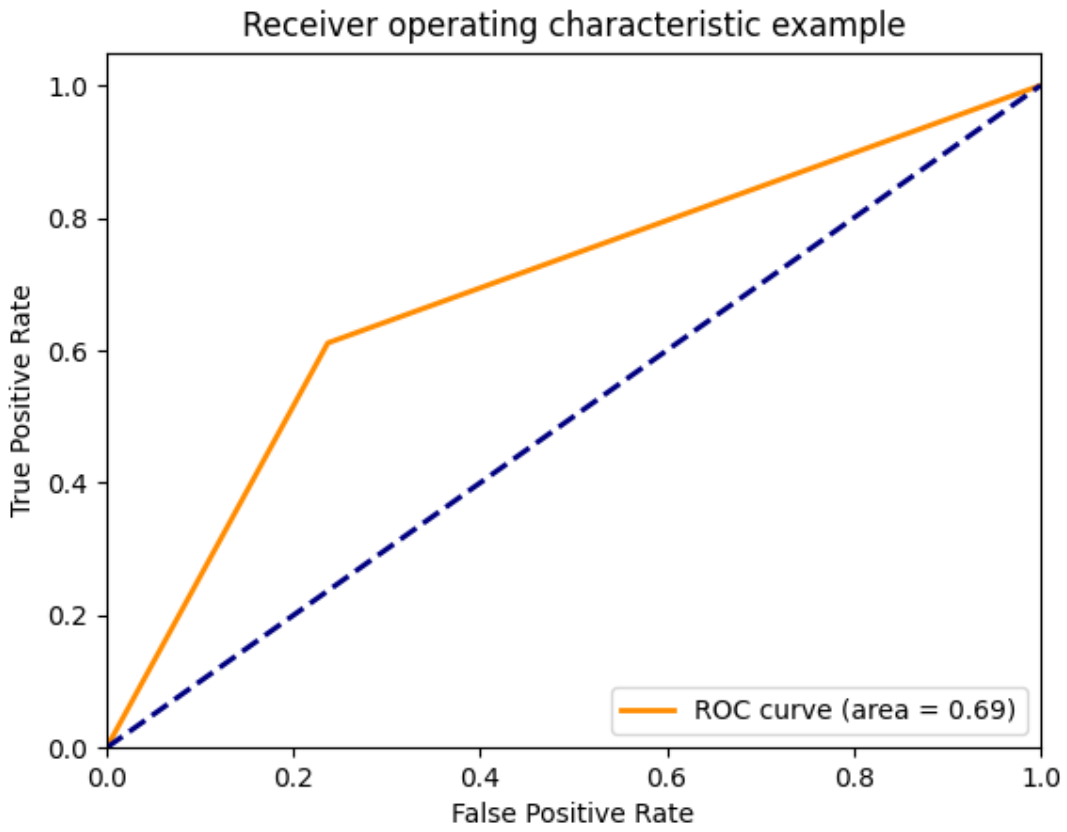
True/False Matrix	True positive	True negative
False positive	TN=34	FP=4
False negative	FN=12	TP=6

This shows us that Naive bayes is better for true negative rate, while decision tree is better for true negative rate. Meaning that Naive bayes had better chance to give a correct diagnostic to a negative case, and the decision tree had better chance for the positive case.

Discuss which method that is most truthful from a scientific point of view, in terms of total accuracy:

Naive bayes was in our case better in terms of accuracy when we tested.
Accuracy Decision Tree: 61.15107913669065
Accuracy Naive Bayes: 72.66187050359713

In the Naive Bayes case, you should visualize this relationship using a ROC curve (Receiver operating characteristic). Explain what conclusions that can be made from the ROC curve:



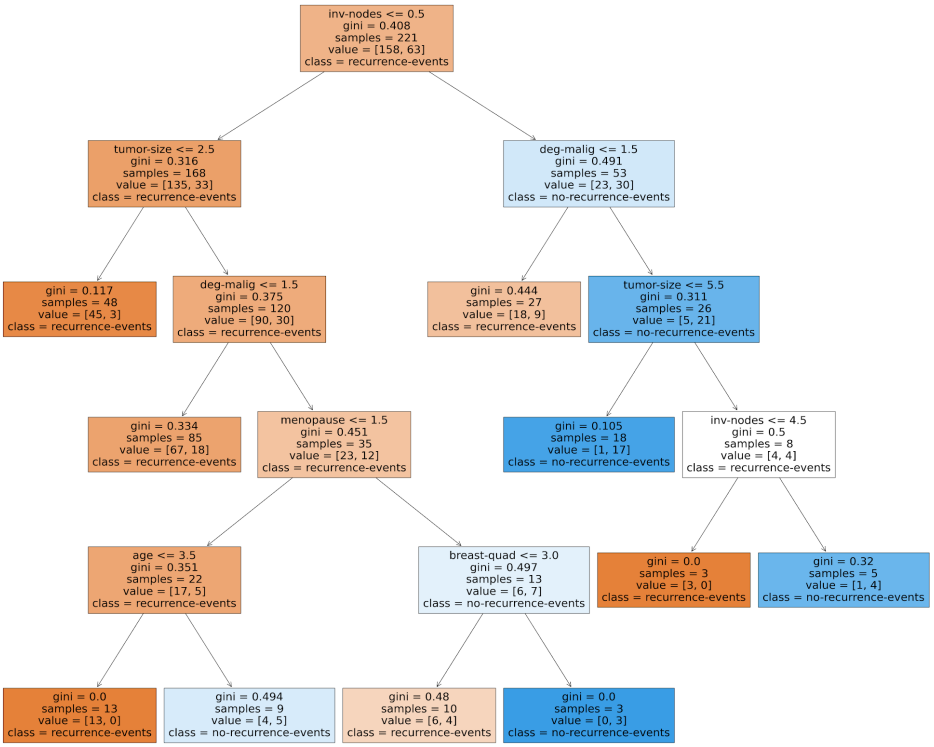
Since the curve is above the blue line it means the model has a good true positive rate.

An analysis on the results and the tree

The results are worse than the naive bayes method. The true positive and true negative rates are a bit low. If you don't prune the tree at all it can result in overfitting, which means it will only give good accuracy for the dataset it trains on and not on new datasets, and the opposite, underfitting, happens when you prune a tree too much so it is too generalized and will have low accuracy.

An analysis on the results when you have applied the Naive Bayes method:

According to the ROC curve our result should be pretty good. The accuracy of Naive Bayes is better then the Decision tree. It also produces good true positive and true negative rates.


inv-nodes <= 0.5
gini = 0.408
samples = 221
value = [158, 63]
class = recurrence-events
tumor-size <= 2.5
gini = 0.316
samples = 168
value = [135, 33]
class = recurrence-events
deg-malig <= 1.5
gini = 0.491
samples = 53
value = [23, 30]
class = no-recurrence-events
gini = 0.117
samples = 48
value = [45, 3]
class = recurrence-events
deg-malig <= 1.5
gini = 0.375
samples = 120
value = [90, 30]
class = recurrence-events
gini = 0.444
samples = 27
value = [18, 9]
class = recurrence-events
tumor-size <= 5.5
gini = 0.311
samples = 26
value = [5, 21]
class = no-recurrence-events
gini = 0.334
samples = 85
value = [67, 18]
class = recurrence-events
menopause <= 1.5
gini = 0.451
samples = 35
value = [23, 12]
class = recurrence-events
gini = 0.105
samples = 18
value = [1, 17]
class = no-recurrence-events
inv-nodes <= 4.5
gini = 0.5
samples = 8
value = [4, 4]
class = recurrence-events
age <= 3.5
gini = 0.351
samples = 22
value = [17, 5]
class = recurrence-events
breast-quad <= 3.0
gini = 0.497
samples = 13
value = [6, 7]
class = no-recurrence-events
gini = 0.0
samples = 3
value = [3, 0]
class = recurrence-events
gini = 0.32
samples = 5
value = [1, 4]
class = no-recurrence-events
gini = 0.0
samples = 13
value = [13, 0]
class = recurrence-events
gini = 0.494
samples = 9
value = [4, 5]
class = no-recurrence-events
gini = 0.48
samples = 10
value = [6, 4]
class = recurrence-events
gini = 0.0
samples = 3
value = [0, 3]
class = no-recurrence-events

Code:

```
from scipy.io import arff
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

# Load the data
dataSet = arff.loadarff("breast-cancer.arff")
# Convert the data to pandas dataframe
df = pd.DataFrame(dataSet[0])
# Go through the dataframe and decode it to remove weird characters
for i in range(0, len(df.columns)):
    title = list(df.columns)[i]
    df[title] = df[title].apply(lambda s: s.decode("utf-8"))

print(df.columns)
# Make a copy of the column names and class names for later
feature_names = list(df.columns)
class_names = df['Class'].unique()

# Remove rows with nan values
df = df[df['node-caps'] != '?']
df = df[df['breast-quad'] != '?']
# Label encode the whole dataframe
df = df.apply(LabelEncoder().fit_transform)

# Remove the Class from the dataframe
Class = df.iloc[:, -1]
df = df[df.columns[:-1]]

# Split the data to a training set and a testing set
x_train, x_test, y_train, y_test = train_test_split(df, Class, test_size=0.2, random_state=0)

# Naive Bayes classification
# Create an instance of Naive Bayes
gnb = GaussianNB()
# Fit the model to the training data and predict the testing data
y_pred = gnb.fit(x_train, y_train).predict(x_test)

# Print out how many wrong classifications it did
```

```
print("Number of mislabeled points out of a total %d points : %d" % (x_test.shape[0], (y_test
!= y_pred).sum()))
# Calculate the accuracy
correct = (y_test == y_pred).sum()
accuracy = correct / len(y_pred)
# Printing the accuracy
print('Accuracy NB:', accuracy * 100)

# Confustion Matrix
names = ['recurr', 'no-recurr']
metrics.plot_confusion_matrix(gnb, x_test, y_test, display_labels=names)

# Decision Tree classification
# Create an instance of a Decision Tree
clf = tree.DecisionTreeClassifier(criterion='gini', ccp_alpha=0.0075)
# Fit the model to the training data and predict the testing data
clf = clf.fit(x_train, y_train)
# Calculate the accuracy
tree_pred = clf.predict(x_test)
corr = (y_test == tree_pred).sum()
accuracy_tree = corr / len(tree_pred)
# Print out how many wrong classifications it did
print("Number of mislabeled points out of a total %d points : %d" % (x_test.shape[0], (y_test
!= tree_pred).sum()))
# Printing the accuracy
print('Accuracy Tree:', accuracy_tree * 100)

# Matrix

metrics.plot_confusion_matrix(clf, x_test, y_test, display_labels=names)

# ROC Curve
fpr = dict()
tpr = dict()
roc_auc = dict()
y_true = y_test.to_numpy()

fpr['NB'], tpr['NB'], _ = roc_curve(y_true, y_pred)
roc_auc['NB'] = auc(fpr['NB'], tpr['NB'])

fpr['micro'], tpr['micro'], _ = roc_curve(y_true.ravel(), y_pred.ravel())
roc_auc['micro'] = auc(fpr['micro'], tpr['micro'])

tnN, fpN, fnN, tpN = metrics.confusion_matrix(y_true, y_pred).ravel()
tnT, fpT, fnT, tpT = metrics.confusion_matrix(y_true, tree_pred).ravel()

# True positive
```

```
true_postitive_NB = tpN / (tpN + fnN)
true_postitive_DT = tpT / (tpT + fnT)

# True negative
true_negative_NB = tnN / (tnN + fpN)
true_negative_DT = tnT / (tnT + fpT)

print('True positive rate NB:\t', str(round(true_postitive_NB * 100, 3)) + '%')
print('True negative rate NB:\t', str(round(true_negative_NB * 100, 3)) + '%')
print('True positive rate DT:\t', str(round(true_postitive_DT * 100, 3)) + '%')
print('True negative rate DT:\t', str(round(true_negative_DT * 100, 3)) + '%')

# Plot the roc curve
plt.figure()
lw = 2
plt.plot(fpr['NB'], tpr['NB'], color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' %
roc_auc['micro'])
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.savefig('roc.png')
plt.show()

# Plotting the Decision Tree
plt.figure()
fig = plt.figure(figsize=(40, 35))
tree.plot_tree(clf, filled=True, feature_names=feature_names, class_names=class_names)
plt.savefig('tree.png')
# plt.show()
```