Assignment 7 Coder

In [1]:
```
from sklearn.linear_model import LogisticRegression
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np

from pylab import rcParams
rcParams['figure.figsize'] = 20, 10

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['font.size'] = 14
from sklearn import preprocessing
enc = preprocessing.OrdinalEncoder()
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

1. Use your own dataset (`Heart.csv` is acceptable), create a train and a test set, and build 2 models: Logistic Regression and Decision Tree (shallow). Compare the test results using `classification_report` and `confusion_matrix`. Explain which algorithm is optimal.

Data Wrangling / Train-Test Split

In [2]:
```
df = pd.read_csv('../data/heart.csv', index_col = 'Unnamed: 0')
df.head
```

Out[2]:
```
<bound method NDFrame.head of      Age  Sex     ChestPain  RestBP  Chol  Fbs  RestECG  MaxHR  Ex
Ang  \
1     63    1       typical     145   233    1        2    150      0
2     67    1  asymptomatic     160   286    0        2    108      1
3     67    1  asymptomatic     120   229    0        2    129      1
4     37    1    nonanginal     130   250    0        0    187      0
5     41    0    nontypical     130   204    0        2    172      0
..   ...  ...           ...     ...   ...  ...      ...    ...    ...
299   45    1       typical     110   264    0        0    132      0
300   68    1  asymptomatic     144   193    1        0    141      0
301   57    1  asymptomatic     130   131    0        0    115      1
302   57    0    nontypical     130   236    0        2    174      0
303   38    1    nonanginal     138   175    0        0    173      0

     Oldpeak  Slope   Ca        Thal  AHD
1        2.3      3  0.0       fixed   No
2        1.5      2  3.0      normal  Yes
3        2.6      2  2.0  reversable  Yes
4        3.5      3  0.0      normal   No
5        1.4      1  0.0      normal   No
..       ...    ...  ...         ...  ...
299      1.2      2  0.0  reversable  Yes
300      3.4      2  2.0  reversable  Yes
301      1.2      2  1.0  reversable  Yes
302      0.0      2  1.0      normal  Yes
303      0.0      1  NaN      normal   No

[303 rows x 14 columns]>
```

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'Fbs', 'RestECG', 'MaxHR',
       'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal', 'AHD'],
      dtype='object')
```

In [4]:
```
df = df.dropna()
```

In [5]:
```
transform_columns = ['ChestPain', 'Thal']
non_num_columns = ['ChestPain', 'Thal', 'Ca']

x = df.copy()
x[transform_columns] = enc.fit_transform(df[transform_columns])

x = pd.concat([x.drop(non_num_columns, axis=1),
              pd.get_dummies(df[transform_columns])], axis=1)

x['AHD']=enc.fit_transform(df[['AHD']])
```

In [6]:
```
x.head()
```

Out[6]:

	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	AHD	ChestPain_asymptomat
1	63	1	145	233	1	2	150	0	2.3	3	0.0	Fal
2	67	1	160	286	0	2	108	1	1.5	2	1.0	Tru
3	67	1	120	229	0	2	129	1	2.6	2	1.0	Tru
4	37	1	130	250	0	0	187	0	3.5	3	0.0	Fal
5	41	0	130	204	0	2	172	0	1.4	1	0.0	Fal

In [7]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x.drop(['AHD'], axis=1), x.AHD, test_size=.2(
x.shape, x_train.shape, x_test.shape
```

Out[7]:
```
((297, 18), (237, 17), (60, 17))
```

Logistic Model

In [8]:
```
logr = LogisticRegression()

logr.fit(preprocessing.scale(x_train), y_train)
```

Out[8]:
```
▾ LogisticRegression
LogisticRegression()
```

In [9]:
```
logpred = logr.predict(preprocessing.scale(x_test))
```

In [10]:
```
accuracy_score(y_test, logpred)
```

Out[10]:
```
0.8166666666666667
```

In [11]:
```
confusion_matrix(y_test, logpred)
```

Out[11]:
```
array([[29,  5],
       [ 6, 20]], dtype=int64)
```

In [12]:
```
print(classification_report(y_test, logpred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.85      0.84        34
         1.0       0.80      0.77      0.78        26

    accuracy                           0.82        60
   macro avg       0.81      0.81      0.81        60
weighted avg       0.82      0.82      0.82        60
```

Decision Tree Model

In [13]:
```
dtc = DecisionTreeClassifier(criterion='entropy', max_depth = 2)

dtc.fit(x_train.values, y_train)
```

Out[13]:

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

In [14]:

```
dtc.tree_.node_count
```

Out[14]:

```
7
```

In [15]:

```
list(zip(x_train.columns, dtc.feature_importances_))
```

Out[15]:

```
[('Age', 0.0),
 ('Sex', 0.0),
 ('RestBP', 0.0),
 ('Chol', 0.0),
 ('Fbs', 0.0),
 ('RestECG', 0.0),
 ('MaxHR', 0.0),
 ('ExAng', 0.20081309417653767),
 ('Oldpeak', 0.0),
 ('Slope', 0.0),
 ('ChestPain_asymptomatic', 0.6082423567772594),
 ('ChestPain_nonanginal', 0.0),
 ('ChestPain_nontypical', 0.0),
 ('ChestPain_typical', 0.0),
 ('Thal_fixed', 0.0),
 ('Thal_normal', 0.19094454904620292),
 ('Thal_reversable', 0.0)]
```

In [16]:

```
dtc_pred = dtc.predict(preprocessing.scale(x_test))
```

In [17]:

```
accuracy_score(y_test, dtc_pred)
```

Out[17]:

```
0.75
```

In [18]:

```
confusion_matrix(y_test, dtc_pred)
```

Out[18]:

```
array([[26,  8],
       [ 7, 19]], dtype=int64)
```

In [19]:

```
print(classification_report(y_test, dtc_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.79      0.76      0.78        34
         1.0       0.70      0.73      0.72        26

    accuracy                           0.75        60
   macro avg       0.75      0.75      0.75        60
weighted avg       0.75      0.75      0.75        60
```

Comparison

In [20]:

```
#logistic regression model
print(classification_report(y_test, logpred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.85      0.84        34
         1.0       0.80      0.77      0.78        26

    accuracy                           0.82        60
   macro avg       0.81      0.81      0.81        60
weighted avg       0.82      0.82      0.82        60
```

In [21]:
```
confusion_matrix(y_test, logpred)
```

Out[21]:
```
array([[29,  5],
       [ 6, 20]], dtype=int64)
```

In [22]:
```
#decision tree model
print(classification_report(y_test, dtc_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.79      0.76      0.78        34
         1.0       0.70      0.73      0.72        26

    accuracy                           0.75        60
   macro avg       0.75      0.75      0.75        60
weighted avg       0.75      0.75      0.75        60
```

In [23]:
```
confusion_matrix(y_test, dtc_pred)
```

Out[23]:
```
array([[26,  8],
       [ 7, 19]], dtype=int64)
```

Which algorithm is optimal?

With a shallow decision tree in this example, the logistic regression model is optimal, although it requires a closer look at the numbers to see why. Looking at the classification reports for both models, we see that the logistic regression has a higher precision and recall on both '0.0' and '1.0' This comes together when evaluating the f1-scores, which are much higher for the logistic regression model. Finally, the overal accuracy score is also much higher for the logistic regression.

It's also important to interpret the confusion matrices, especially when dealing with a dataset used to predict heart disease. Ideally we'd minimize both false positives (predicted to have heart disease but actually didn't) as well as false negatives (predicted to not have heart disease, but actually does). However, the consequences of false negatives are severe, in that the individual would not be recommended a treatment program which they need to combat a substantial health risk. As such, we should also prioritize a model that limits false negatives. Looking simply at the numbers, the logistic regression model only had 6 false negatives, while the decision tree had 7. This is reflected in the higher recall on '1.0' in the logistic regression model. Given the small test dataset, the recall is relatively close for both models, but given the classification report comparison coupled with less false negatives, we can conclude the logistic regression model is optimal.

One interesting note is looking at feature importances in the decision tree model. With so few nodes (only 7), the decision tree model is relying heavily on only three features ('ExAng', 'ChestPain' and 'Thal') when predicting an outcome. I would expect this to change as the Decision Tree becomes much deeper.

2. Repeat 1. but let the Decision Tree be much deeper to allow over-fitting. Compare the two models' test results again, and explain which is optimal

In [24]:
```
dtcd = DecisionTreeClassifier(criterion='entropy', max_depth = 10)

dtcd.fit(x_train.values, y_train)
```

Out[24]:
```
DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=10)
```

In [25]:
```
dtcd.tree_.node_count
```

Out[25]:
```
77
```

In [26]:
```
list(zip(x_train.columns, dtcd.feature_importances_))
```

Out[26]:
```
[('Age', 0.03888737255688759),
 ('Sex', 0.013732164119706004),
 ('RestBP', 0.163652658676815),
 ('Chol', 0.08504697920080527),
 ('Fbs', 0.0),
 ('RestECG', 0.0),
 ('MaxHR', 0.10311757560642501),
 ('ExAng', 0.06508542797895153),
 ('Oldpeak', 0.17933678562548194),
 ('Slope', 0.022195454146699545),
 ('ChestPain_asymptomatic', 0.19713711532661274),
 ('ChestPain_nonanginal', 0.0),
 ('ChestPain_nontypical', 0.0),
 ('ChestPain_typical', 0.015806622278137394),
 ('Thal_fixed', 0.013040624555166138),
 ('Thal_normal', 0.0618869389263762),
 ('Thal_reversable', 0.04107428100193567)]
```

In [27]:
```
dtcd_pred = dtcd.predict(preprocessing.scale(x_test))
```

In [28]:
```
accuracy_score(y_test, dtcd_pred)
```

Out[28]:
```
0.4666666666666667
```

Compare the models again

In [29]:
```
#Deep Decision Tree
print(classification_report(y_test, dtcd_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.67      0.12      0.20        34
         1.0       0.44      0.92      0.60        26

    accuracy                           0.47        60
   macro avg       0.56      0.52      0.40        60
weighted avg       0.57      0.47      0.37        60
```

In [30]:
```
confusion_matrix(y_test, dtcd_pred)
```

Out[30]:
```
array([[ 4, 30],
       [ 2, 24]], dtype=int64)
```

In [31]:
```
#Log Regression (for reminder)
print(classification_report(y_test, logpred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.85      0.84        34
         1.0       0.80      0.77      0.78        26

    accuracy                           0.82        60
   macro avg       0.81      0.81      0.81        60
weighted avg       0.82      0.82      0.82        60
```

In [32]:
```
confusion_matrix(y_test, logpred)
```

Out[32]:
```
array([[29,  5],
       [ 6, 20]], dtype=int64)
```

Which algorithm is optimal?

In the case of a deep decision tree which is overfit, it is even more clear that the logistic regression model is optimal. We can see that a decision tree with max depth of 10 leads to 77 nodes (which is more nodes than data points in the test set, which is 60). The decision tree accuracy is only 47%, worse than the flip of a coin whether the model correctly classifies someone as having heart disease or not. We can look further into the classification matrices and see that the logistic model outperforms the over-fitted decision tree in all categories except 1, recall on '1.0', leading us to the conclusion that the logistic model as the optimal choice in this case.

However, an interesting discussion point arises when considering extremely high recall of the decision tree model on 1.0. In this case, there are very few individuals who actually have heart disease who are told that they do not. The user of this model would need to consider the side effects of the treatment and any potential risks of treating someone for heart disease if they don't have it (extremely low recall on '0.0'). If the side effects were minimal, one might actually choose this overfit model due to its extremely high recall on '1.0'. In other words, very few people would 'fall through the cracks,' allowing the highest true positive rate of any model offerred thus far.

As mentioned in question 1, we do see the decision tree model now considering nearly all of the variables in the test set, with some variables actually surpassing the importance of 'ChestPain'. However, the model is simply overfit, putting too much importance on some of these variables and leading to lower accuracy.

In [33]:
```
##### For curiosity's sake, decision tree with depth 3.
dtc3 = DecisionTreeClassifier(criterion='entropy', max_depth = 3)

dtc3.fit(x_train.values, y_train)

dtc3.tree_.node_count
```

Out[33]:
```
15
```

In [34]:
```
list(zip(x_train.columns, dtc3.feature_importances_))
```

Out[34]:
```
[('Age', 0.0),
 ('Sex', 0.0),
 ('RestBP', 0.0),
 ('Chol', 0.0),
 ('Fbs', 0.0),
 ('RestECG', 0.0),
 ('MaxHR', 0.06121194429713762),
 ('ExAng', 0.14328453559098092),
 ('Oldpeak', 0.1348418031368576),
 ('Slope', 0.0),
 ('ChestPain_asymptomatic', 0.43399422719405445),
 ('ChestPain_nonanginal', 0.0),
 ('ChestPain_nontypical', 0.0),
 ('ChestPain_typical', 0.0),
 ('Thal_fixed', 0.0),
 ('Thal_normal', 0.136243112760677),
 ('Thal_reversable', 0.09042437702029245)]
```

In [35]:
```
dtc3_pred = dtc3.predict(preprocessing.scale(x_test))
```

In [36]:
```
print(classification_report(y_test, dtc3_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.79      0.76      0.78        34
         1.0       0.70      0.73      0.72        26

    accuracy                           0.75        60
   macro avg       0.75      0.75      0.75        60
weighted avg       0.75      0.75      0.75        60
```

In [37]:
```
confusion_matrix(y_test, dtc3_pred)
```

Out[37]:
```
array([[26,  8],
       [ 7, 19]], dtype=int64)
```

Identical results to the original model, but not considering more variables in the decision tree.

In [43]:
```
##### For curiosity's sake, decision tree with depth 6.
dtc6 = DecisionTreeClassifier(criterion='entropy', max_depth = 6)

dtc6.fit(x_train.values, y_train)

dtc6.tree_.node_count
```

Out[43]:
```
61
```

In [44]:
```
list(zip(x_train.columns, dtc6.feature_importances_))
```

Out[44]:
```
[('Age', 0.03124673648709512),
 ('Sex', 0.0),
 ('RestBP', 0.12716578043453988),
 ('Chol', 0.08986077287640729),
 ('Fbs', 0.0),
 ('RestECG', 0.0),
 ('MaxHR', 0.14971677269555908),
 ('ExAng', 0.08806470233500523),
 ('Oldpeak', 0.12071378673801123),
 ('Slope', 0.0157580083523728),
 ('ChestPain_asymptomatic', 0.2262198647496176),
 ('ChestPain_nonanginal', 0.0),
 ('ChestPain_nontypical', 0.0),
 ('ChestPain_typical', 0.018138501965924928),
 ('Thal_fixed', 0.014964449075115288),
 ('Thal_normal', 0.07101683988069758),
 ('Thal_reversable', 0.047133784409654066)]
```

In [45]:
```
dtc6_pred = dtc6.predict(preprocessing.scale(x_test))
print(classification_report(y_test, dtc6_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.50      0.06      0.11        34
         1.0       0.43      0.92      0.59        26

    accuracy                           0.43        60
   macro avg       0.46      0.49      0.35        60
weighted avg       0.47      0.43      0.31        60
```

Essentially the same as the depth=10 model. Overfit now (61 nodes with 60 data points), again extremely high recall on '1.0' but low everything else (including only 6% recall on '0.0'!)

In [ ]: