Assignment is at the bottom!

In [ ]:
```
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


from sklearn.linear_model import LogisticRegression as Model
```

In [ ]:
```
y = np.concatenate([np.zeros(10), np.ones(10)])
x = np.linspace(0, 10, len(y))
```

In [ ]:
```
plt.scatter(x, y, c=y)
```

In [ ]:
```
model = LogisticRegression()
```

In [ ]:
```
model.fit(x.reshape(-1, 1),y)
```

In [ ]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1))[:,1])
```

In [ ]:
```
b, b0 = model.coef_, model.intercept_
model.coef_, model.intercept_
```

In [ ]:
```
plt.plot(x, 1/(1+np.exp(-x)))
```

In [ ]:
```
b
```

In [ ]:
```
plt.plot(x, 1/(1+np.exp(-(b[0]*x +b0))))
```

In [ ]:
```
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import

import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np


fig = plt.figure()
ax = fig.gca(projection='3d')

# Make data.
X = np.arange(-10, 10, 0.25)
Y = np.arange(-10, 10, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = 1/(1+np.exp(-(b[0]*X +b[0]*Y +b0)))
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
```

In [ ]:
```
X
```

In [ ]:
```
Y
```

What if the data doesn't really fit this pattern?

In [ ]:
```
y = np.concatenate([np.zeros(10), np.ones(10), np.zeros(10)])
x = np.linspace(0, 10, len(y))
```

In [ ]:
```
plt.scatter(x,y, c=y)
```

In [ ]:
```
model.fit(x.reshape(-1, 1),y)
```

In [ ]:
```
plt.scatter(x,y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1)))
```

In [ ]:
```
model1 = LogisticRegression()
model1.fit(x[:15].reshape(-1, 1),y[:15])
```

In [ ]:
```
model2 = LogisticRegression()
model2.fit(x[15:].reshape(-1, 1),y[15:])
```

In [ ]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model1.predict_proba(x.reshape(-1, 1))[:,1] * model2.predict_proba(x.reshape(-1, 1))[:,1])
```

In [ ]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [ ]:
```
from sklearn import preprocessing

enc = preprocessing.OrdinalEncoder()
```

In [ ]:
```
transform_columns = ['sex', 'workclass', 'education', 'marital-status',
                     'occupation', 'relationship', 'race', 'sex',
                     'native-country', 'salary']
```

In [ ]:
```
x = df.copy()

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

golden['salary'] = golden.salary.replace(' <=50K.', ' <=50K').replace(' >50K.', ' >50K')
xt = golden.copy()

xt[transform_columns] = enc.transform(golden[transform_columns])
```

In [ ]:
```
df.salary.unique()
```

In [ ]:
```
golden.salary.replace(' <=50K.', ' <=50K').replace(' >50K.', ' >50K').unique()
```

In [ ]:
```
model.fit(preprocessing.scale(x.drop('salary', axis=1)), x.salary)
```

In [ ]:
```
pred = model.predict(preprocessing.scale(x.drop('salary', axis=1)))
pred_test = model.predict(preprocessing.scale(xt.drop('salary', axis=1)))
```

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

In [ ]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [ ]:
```
accuracy_score(x.salary, pred)
```

In [ ]:
```
confusion_matrix(x.salary, pred)
```

In [ ]:
```
print(classification_report(x.salary, pred))
```

In [ ]:
```
print(classification_report(xt.salary, pred_test))
```

Assignment

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

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

ANSWER 1

In [1]:
```
import pandas as pd
from sklearn import preprocessing
```

In [2]:
```
heart = pd.read_csv('../data/Heart.csv')
```

In [3]:
```
heart.head()
```

Out[3]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

In [4]:
```
transform_cols = ['ChestPain', 'Thal', 'AHD']

enc = preprocessing.OrdinalEncoder()
```

In [5]:
```
heart[transform_cols] = enc.fit_transform(heart[transform_cols])
```

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

Out[6]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	3.0	145	233	1	2	150	0	2.3	3	0.0	0.0	0.0
1	2	67	1	0.0	160	286	0	2	108	1	1.5	2	3.0	1.0	1.0
2	3	67	1	0.0	120	229	0	2	129	1	2.6	2	2.0	2.0	1.0
3	4	37	1	1.0	130	250	0	0	187	0	3.5	3	0.0	1.0	0.0
4	5	41	0	2.0	130	204	0	2	172	0	1.4	1	0.0	1.0	0.0

In [7]:
```
print(heart.isna().sum().sum())
print(len(heart))
```

```
6
303
```

In [8]:
```
heart = heart.dropna()
print(heart.isna().sum().sum())
print(len(heart))
```

```
0
297
```

In [9]:
```
from sklearn.linear_model import LogisticRegression as LOGmodel
from sklearn.tree import DecisionTreeClassifier as DTCmodel

from sklearn.metrics import (accuracy_score,
                             classification_report,
                             confusion_matrix, auc, roc_curve
                             )
from sklearn.metrics import *
from sklearn import model_selection
```

In [10]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(heart.drop('AHD', axis=1),
                                                                    heart.AHD, test_size=0.2, random_state=0)
```

In [11]:
```
LOGmodel = LOGmodel()
DTCmodel = DTCmodel(criterion='entropy', max_depth=2)
```

In [12]:
```
LOGmodel.fit(preprocessing.scale(X_train), y_train)
DTCmodel.fit(X_train, y_train)
```

Out[12]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

In [13]:
```
LOG_predict = LOGmodel.predict(preprocessing.scale(X_test))
DTC_predict = DTCmodel.predict(X_test)
```

In [14]:
```
confusion_matrix(y_test, LOG_predict)
```

Out[14]:
```
array([[27,  3],
       [11, 19]])
```

In [15]:
```
print(classification_report(y_test, LOG_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.71      0.90      0.79        30
         1.0       0.86      0.63      0.73        30

    accuracy                           0.77        60
   macro avg       0.79      0.77      0.76        60
weighted avg       0.79      0.77      0.76        60
```

In [16]:
```
confusion_matrix(y_test, DTC_predict)
```

Out[16]:
```
array([[21,  9],
       [12, 18]])
```

In [17]:
```
print(classification_report(y_test, DTC_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.64      0.70      0.67        30
         1.0       0.67      0.60      0.63        30

    accuracy                           0.65        60
   macro avg       0.65      0.65      0.65        60
weighted avg       0.65      0.65      0.65        60
```

When comparing the logistic and shallow decision tree models, the logistic model is optimal. The confusion matrix for the logistic model predicted significantly more true negatives, and one more true positive, than the shallow decision tree model. The confusion matrix values are refelcted in the classification reports as well (i.e. recall for true negatives significantly higher in logisitic model and higher f1 scores overall).

In [ ]:

Answer 2

In [18]:
```
from sklearn.tree import DecisionTreeClassifier as DTCmodel

DTCmodel2 = DTCmodel(criterion='entropy', max_depth=None)
```

In [19]:
```
DTCmodel2.fit(X_train, y_train)
```

Out[19]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [20]:
```
DTC_predict2 = DTCmodel2.predict(X_test)
```

In [21]:
```
DTCmodel2.get_depth()
```

Out[21]:
```
7
```

In [22]:
```
print(accuracy_score(y_test, DTC_predict), accuracy_score(y_test, DTC_predict2), accuracy_score(y_test, LOG_predict))
```

```
0.65 0.7166666666666667 0.7666666666666667
```

In [23]:
```
confusion_matrix(y_test, DTC_predict2)
```

Out[23]:
```
array([[23,  7],
       [10, 20]])
```

In [24]:
```
confusion_matrix(y_test, DTC_predict)
```

Out[24]:
```
array([[21,  9],
       [12, 18]])
```

In [25]:
```
confusion_matrix(y_test, LOG_predict)
```

Out[25]:
```
array([[27,  3],
       [11, 19]])
```

In [26]:
```
print(classification_report(y_test, DTC_predict2))
```

```
              precision    recall  f1-score   support

         0.0       0.70      0.77      0.73        30
         1.0       0.74      0.67      0.70        30

    accuracy                           0.72        60
   macro avg       0.72      0.72      0.72        60
weighted avg       0.72      0.72      0.72        60
```

In [27]:
```
print(classification_report(y_test, DTC_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.64      0.70      0.67        30
         1.0       0.67      0.60      0.63        30

    accuracy                           0.65        60
   macro avg       0.65      0.65      0.65        60
weighted avg       0.65      0.65      0.65        60
```

In [28]:
```
print(classification_report(y_test, LOG_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.71      0.90      0.79        30
         1.0       0.86      0.63      0.73        30

    accuracy                           0.77        60
   macro avg       0.79      0.77      0.76        60
weighted avg       0.79      0.77      0.76        60
```

With no limits placed on the decision tree, the depth maxed out at 7. This led to better results than the orginal tree model, but inferior results compared to the logistic model. The true positive rate was higher for the maxed out decision tree, but the logitsic model was much better at predicting true negatives. The logistic model aslo performs better than the other models when taking into account the f1-scores. Overall, the logistic model is still optimal.

In [ ]: