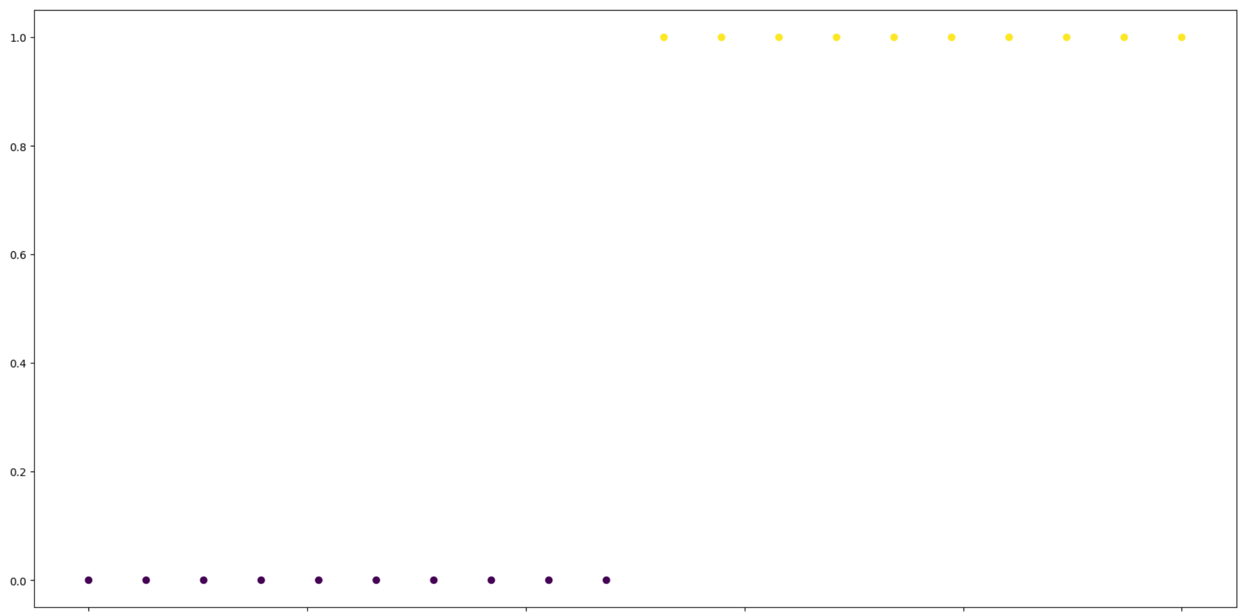
Assignment is at the bottom!

In [2]:
```
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 [3]:
```
y = np.concatenate([np.zeros(10), np.ones(10)])
x = np.linspace(0, 10, len(y))
```

In [4]:
```
plt.scatter(x, y, c=y)
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0x7f8d695c1870>
```



In [5]:
```
model = LogisticRegression()
```

In [6]:
```
model.fit(x.reshape(-1, 1),y)
```

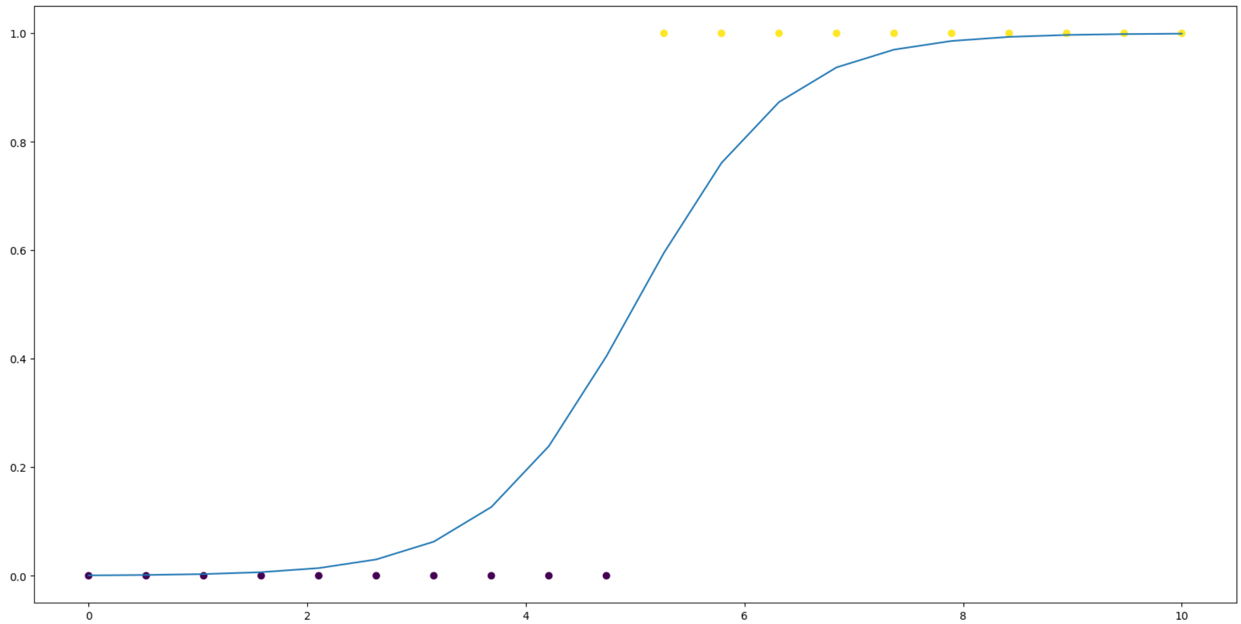
Out[6]:

▾ LogisticRegression

LogisticRegression()

In [7]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1))[:,1])
```

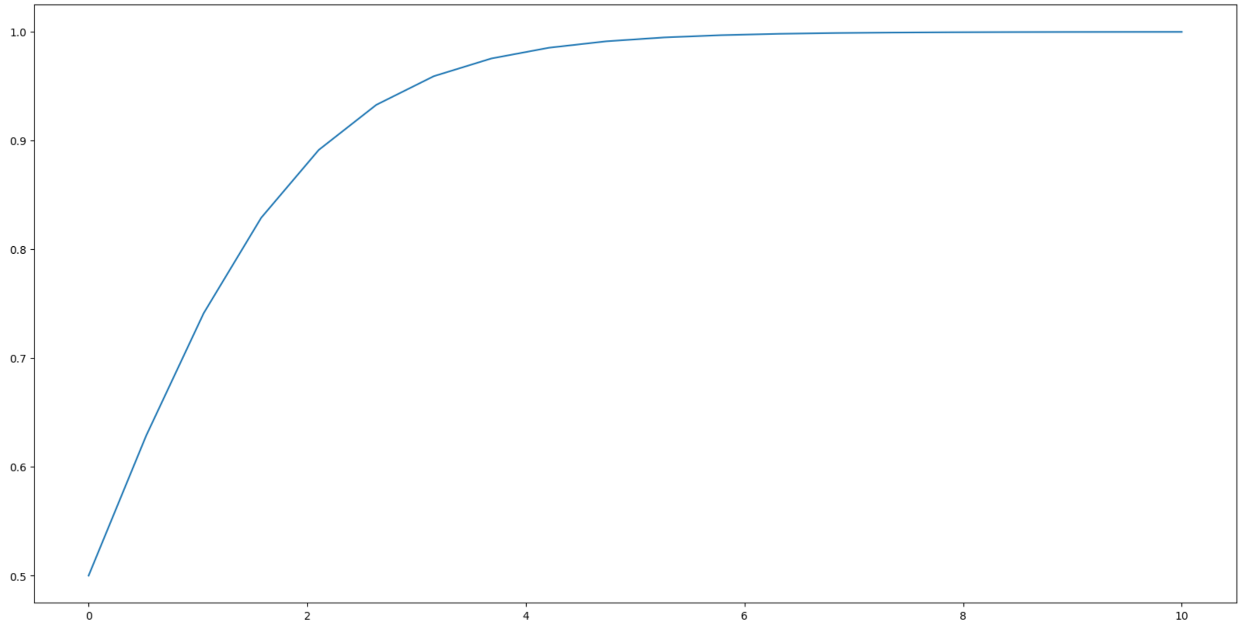
Out[7]: [<matplotlib.lines.Line2D at 0x7f8d69ea28c0>]



In [8]:
```
b, b0 = model.coef_, model.intercept_
model.coef_, model.intercept_
```

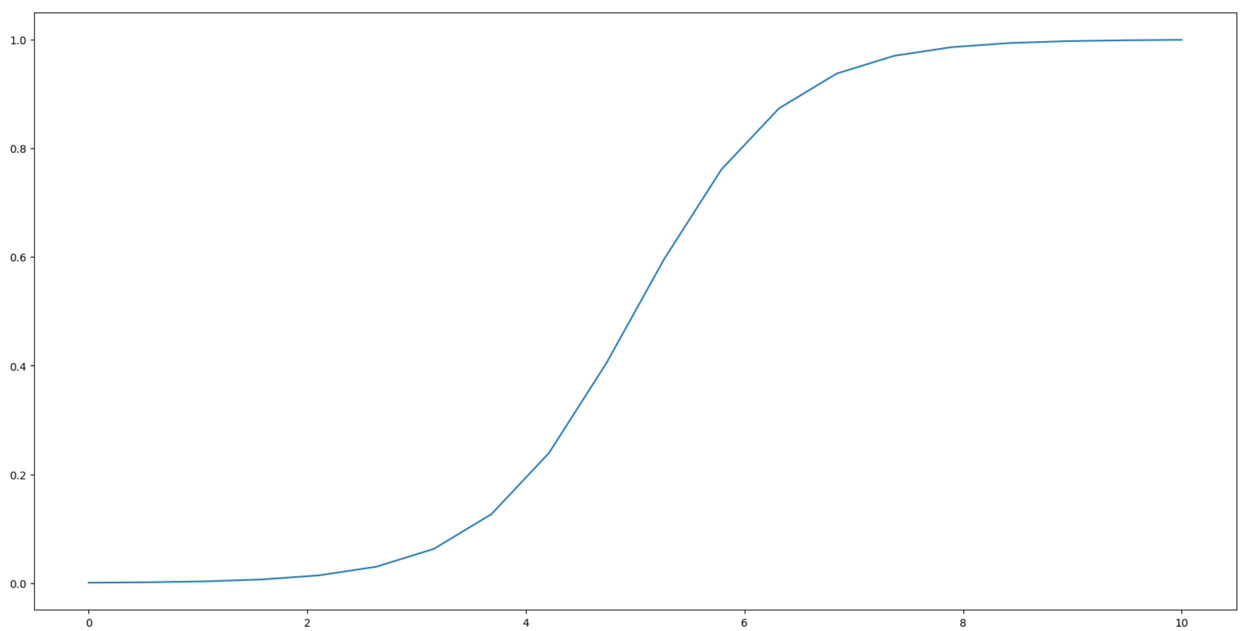
Out[8]: (array([[1.46709085]]), array([-7.33542562]))

In [9]:
```
plt.plot(x, 1/(1+np.exp(-x)))
```
Out[9]: [<matplotlib.lines.Line2D at 0x7f8d6a947ac0>]



In [10]:
```
b
```
Out[10]: array([[1.46709085]])

In [11]:
```
plt.plot(x, 1/(1+np.exp(-(b[0]*x +b0))))
```
Out[11]: [<matplotlib.lines.Line2D at 0x7f8d6a9d10f0>]



In [12]:
```
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')# Doesn't like "projection"

# 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)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[12], line 10
      6 import numpy as np
      9 fig = plt.figure()
---> 10 ax = fig.gca(projection='3d')
     12 # Make data.
     13 X = np.arange(-10, 10, 0.25)

TypeError: FigureBase.gca() got an unexpected keyword argument 'projection'


<Figure size 2000x1000 with 0 Axes>
```

In [13]:
```
X
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 X

NameError: name 'X' is not defined
```

In [13]:
```
Y
```

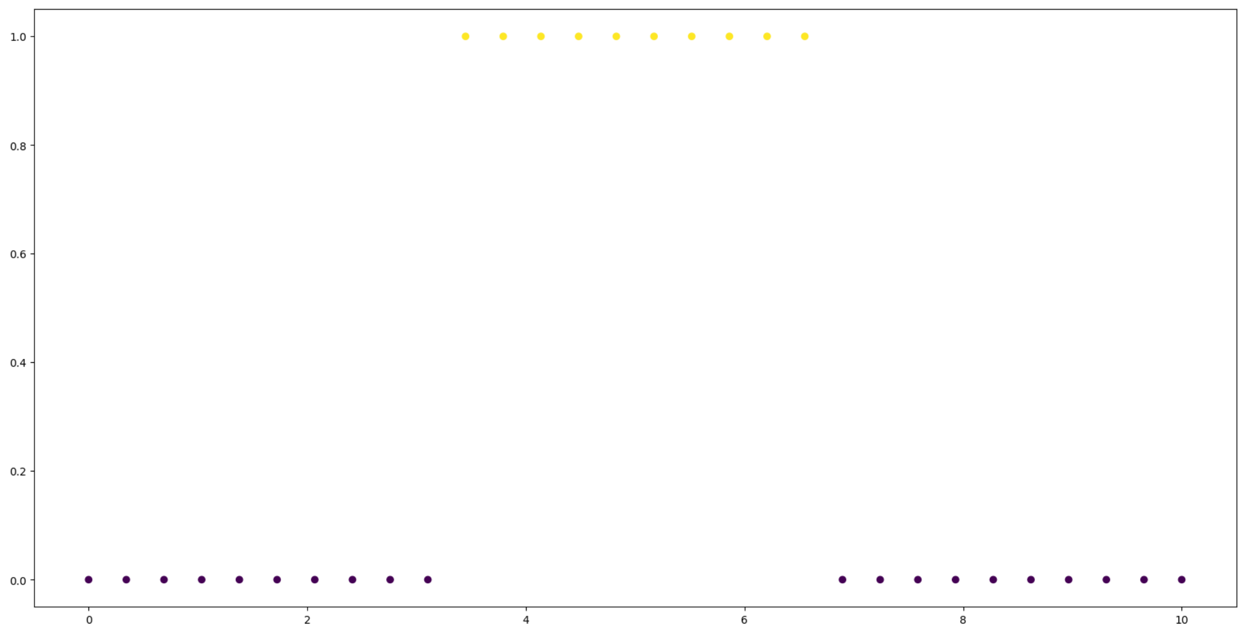
Out[13]:
```
array([[-10.  , -10.  , -10.  , ..., -10.  , -10.  , -10.  ],
       [ -9.75,  -9.75,  -9.75, ...,  -9.75,  -9.75,  -9.75],
       [ -9.5 ,  -9.5 ,  -9.5 , ...,  -9.5 ,  -9.5 ,  -9.5 ],
       ...,
       [  9.25,   9.25,   9.25, ...,   9.25,   9.25,   9.25],
       [  9.5 ,   9.5 ,   9.5 , ...,   9.5 ,   9.5 ,   9.5 ],
       [  9.75,   9.75,   9.75, ...,   9.75,   9.75,   9.75]])
```

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

In [14]:
```
y = np.concatenate([np.zeros(10), np.ones(10), np.zeros(10)])
x = np.linspace(0, 10, len(y))
```

In [15]:
```
plt.scatter(x,y, c=y)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x7f8d6b518670>
```

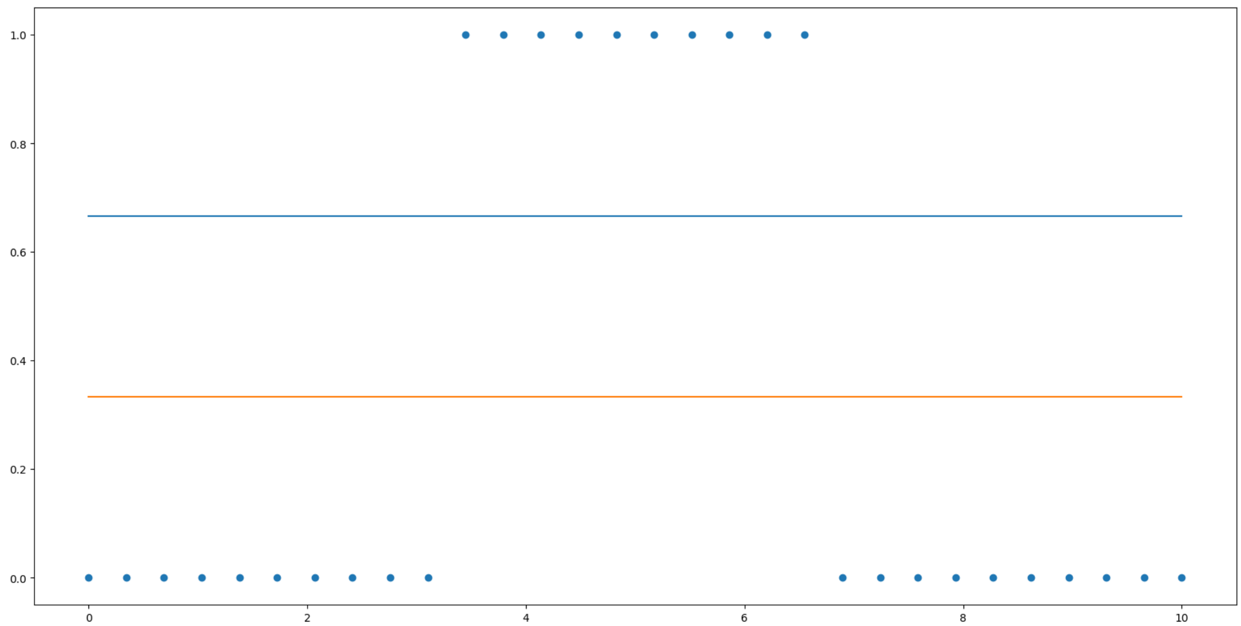


In [16]:
```
model.fit(x.reshape(-1, 1),y)
```

Out[16]:
```
▾ LogisticRegression
LogisticRegression()
```

In [17]:
```
plt.scatter(x,y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1)))
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x7f8d6b533e20>,
 <matplotlib.lines.Line2D at 0x7f8d6b533e80>]
```



In [18]:
```
model1 = LogisticRegression()
model1.fit(x[:15].reshape(-1, 1),y[:15])
```

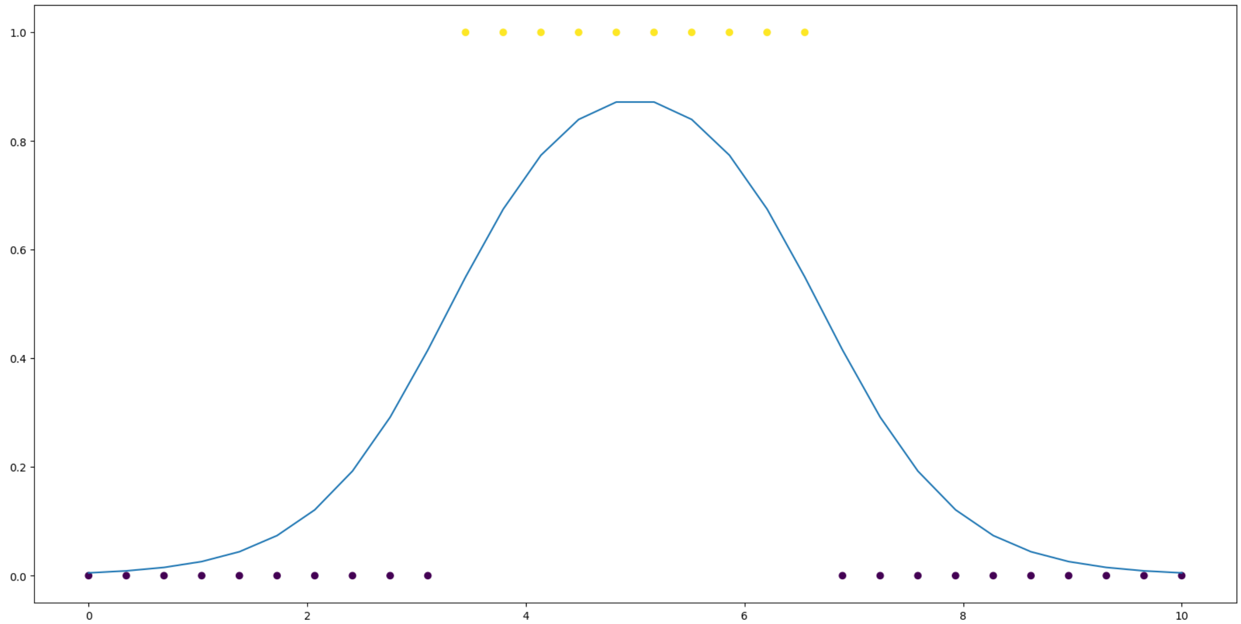
Out[18]:
```
▾ LogisticRegression
LogisticRegression()
```

In [19]:
```
model2 = LogisticRegression()
model2.fit(x[15:].reshape(-1, 1),y[15:])
```

Out[19]:
```
▾ LogisticRegression
LogisticRegression()
```

In [20]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model1.predict_proba(x.reshape(-1, 1))[:,1] * model2.predi
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x7f8d6c2ab910>]
```



In [21]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [22]:
```
from sklearn import preprocessing

enc = preprocessing.OrdinalEncoder()
```

In [23]:
```
transform_columns = ['sex', 'workclass', 'education', 'marital-status'
                     'occupation', 'relationship', 'race', 'sex',
                     'native-country', 'salary']
```

In [24]:
```
x = df.copy()

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

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

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

In [25]:
```
df.salary.unique()
```

Out[25]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [26]:
```
golden.salary.replace(' <=50K.', ' <=50K').replace(' >50K.', ' >50K').
```

Out[26]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [27]:
```
model.fit(preprocessing.scale(x.drop('salary', axis=1)), x.salary)
```

Out[27]:
```
▾ LogisticRegression
LogisticRegression()
```

In [28]:
```
pred = model.predict(preprocessing.scale(x.drop('salary', axis=1)))
pred_test = model.predict(preprocessing.scale(xt.drop('salary', axis=1
```

In [29]:
```
x.head()
```

Out[29]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex
0	39	7.0	77516	9.0	13	4.0	1.0	1.0	4.0	1.(
1	50	6.0	83311	9.0	13	2.0	4.0	0.0	4.0	1.(
2	38	4.0	215646	11.0	9	0.0	6.0	1.0	4.0	1.(
3	53	4.0	234721	1.0	7	2.0	6.0	0.0	2.0	1.(
4	28	4.0	338409	9.0	13	2.0	10.0	5.0	2.0	0.(

In [30]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [31]:
```
accuracy_score(x.salary, pred)
```

Out[31]:
```
0.8250360861152913
```

In [32]:
```
confusion_matrix(x.salary, pred)
```

Out[32]:
```
array([[23300,  1420],
       [ 4277,  3564]])
```

In [33]:
```
print(classification_report(x.salary, pred))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.94      0.89     24720
         1.0       0.72      0.45      0.56      7841

    accuracy                           0.83     32561
   macro avg       0.78      0.70      0.72     32561
weighted avg       0.81      0.83      0.81     32561
```

In [34]:
```
print(classification_report(xt.salary, pred_test))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.94      0.89     12435
         1.0       0.70      0.45      0.55      3846

    accuracy                           0.82     16281
   macro avg       0.77      0.69      0.72     16281
weighted avg       0.81      0.82      0.81     16281
```

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

In [48]:
```
# Data load

heart = pd.read_csv('../data/Heart.csv', index_col=False)
heart.head()
```

Out[48]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeal
0	1	63	1	typical	145	233	1	2	150	0	2.3
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5
4	5	41	0	nontypical	130	204	0	2	172	0	1.4

In [49]:
```
# Data Transform and Cleaning

heart_transform_clmns = ['ChestPain', 'Thal']

heart_clean = heart.dropna()

ht = heart_clean.copy()

ht[heart_transform_clmns] = enc.fit_transform(heart_clean[heart_transf
```

In [138]:
```
# Model Definition

from sklearn.tree import DecisionTreeClassifier

logi = LogisticRegression()
short_tree = DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

In [127]:
```
# Train Test Split

from sklearn.model_selection import train_test_split

ht_x_train, ht_x_test, ht_y_train, ht_y_test = train_test_split(ht.drc
                                                    ht.AHD, test_size=
```

In [139]:
```
# Model Fit

logi.fit(preprocessing.scale(ht_x_train.values), ht_y_train)
short_tree.fit(preprocessing.scale(ht_x_train.values), ht_y_train)
```

Out[139]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

In [140]:
```
# Testing

logi_predictions = logi.predict(preprocessing.scale(ht_x_test.values))
short_tree_predictions = short_tree.predict(preprocessing.scale(ht_x_t
```

In [141]:
```
# Accuracy

accuracy_score(ht_y_test, logi_predictions),accuracy_score(ht_y_test,
```

Out[141]:
```
(0.9, 0.8833333333333333)
```

In [142]:
```
# Confusion Matrix

confusion_matrix(ht_y_test, logi_predictions),confusion_matrix(ht_y_te
```

Out[142]:
```
(array([[27,  2],
        [ 4, 27]]),
 array([[25,  4],
        [ 3, 28]]))
```

In [143]:
```
# Classification Report

print(classification_report(ht_y_test, logi_predictions),classificatic
```

```
              precision    recall  f1-score   support

          No       0.87      0.93      0.90        29
         Yes       0.93      0.87      0.90        31

    accuracy                           0.90        60
   macro avg       0.90      0.90      0.90        60
weighted avg       0.90      0.90      0.90        60
               precision    recall  f1-score   support

          No       0.89      0.86      0.88        29
         Yes       0.88      0.90      0.89        31

    accuracy                           0.88        60
   macro avg       0.88      0.88      0.88        60
weighted avg       0.88      0.88      0.88        60
```

Results Disucssion:

First off I would say that the overall results regarding the accuracy, confusion matrix, and classification reports vary wildly depending on what the test train split uses for the training data. I was seeing the accuracy and subsequent report values varying from 0.50 to above 0.90. I used the current test train split as I think it is fairly representative of the general trends between the two models. Broadly speaking with this data set the logistic regression model performed roughly equal to or, slightly better than, the shallow decision tree. The trend that appeared to be holding the shallow decision tree back was overall inferior precision while generally having higher recall. Between the two models it is a close call but ultimately I would choose the logistic regression for it's more conistent performance.

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 [144]:
```
# Model Definition

tall_tree = DecisionTreeClassifier(criterion='entropy', max_depth=15)
```

In [145]:
```
# Model Fit

tall_tree.fit(preprocessing.scale(ht_x_train.values), ht_y_train)
```

Out[145]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=15)
```

In [146]:
```
# Testing

tall_tree_predictions = tall_tree.predict(preprocessing.scale(ht_x_tes
```

In [147]:
```
# Accuracy

accuracy_score(ht_y_test, logi_predictions),accuracy_score(ht_y_test,
```

Out[147]:
```
(0.9, 0.7)
```

In [148]:
```
# Confusion Matrix

confusion_matrix(ht_y_test, logi_predictions),confusion_matrix(ht_y_te
```

Out[148]:
```
(array([[27,  2],
        [ 4, 27]]),
 array([[19, 10],
        [ 8, 23]]))
```

In [149]:
```
# Classification Report

print(classification_report(ht_y_test, logi_predictions),classificatic
```

```
              precision    recall  f1-score   support

          No       0.87      0.93      0.90        29
         Yes       0.93      0.87      0.90        31

    accuracy                           0.90        60
   macro avg       0.90      0.90      0.90        60
weighted avg       0.90      0.90      0.90        60
               precision    recall  f1-score   support

          No       0.70      0.66      0.68        29
         Yes       0.70      0.74      0.72        31

    accuracy                           0.70        60
   macro avg       0.70      0.70      0.70        60
weighted avg       0.70      0.70      0.70        60
```

Results Disucssion:

As mentioned above the actual values for the accuracy, confusion matrix, and the classification report vary significantly depending on the train test split data. The performance between the two models remained fairly consistent between the training sets. Here we can clearly see the deep decision tree overfitting for the training set. This is evident in the low accuracy values, amount of missed estimates in the confusion matrix, and the precision, recall, and f1 scores, when compared to the logistic regression model from earlier. Between the two models the logistic regression is clearly more consitently capable at prediction with this data set.