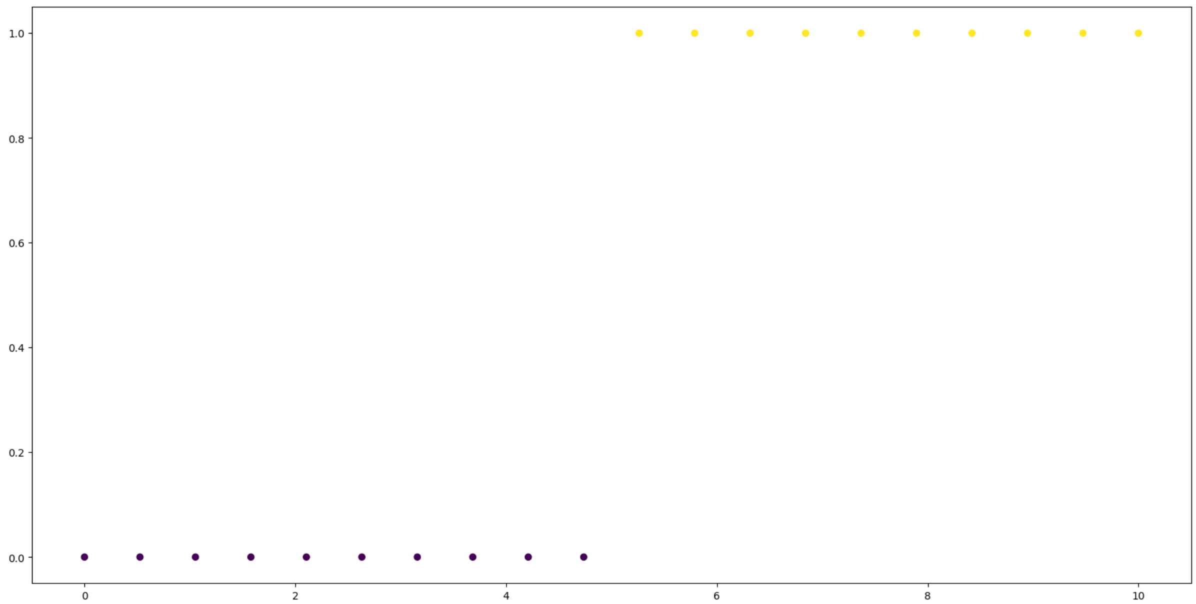
Assignment is at the bottom!

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 0x7ff199437ee0>
```



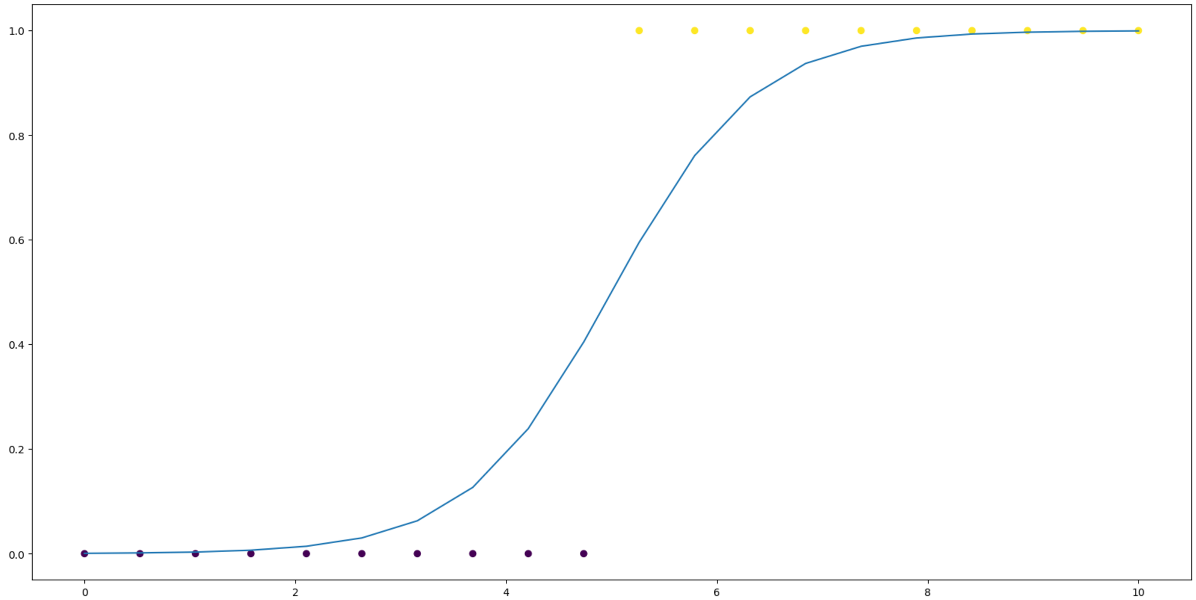
In [5]:
```
model = LogisticRegression()
```

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

Out[6]:
```
▾ LogisticRegression
LogisticRegression()
```

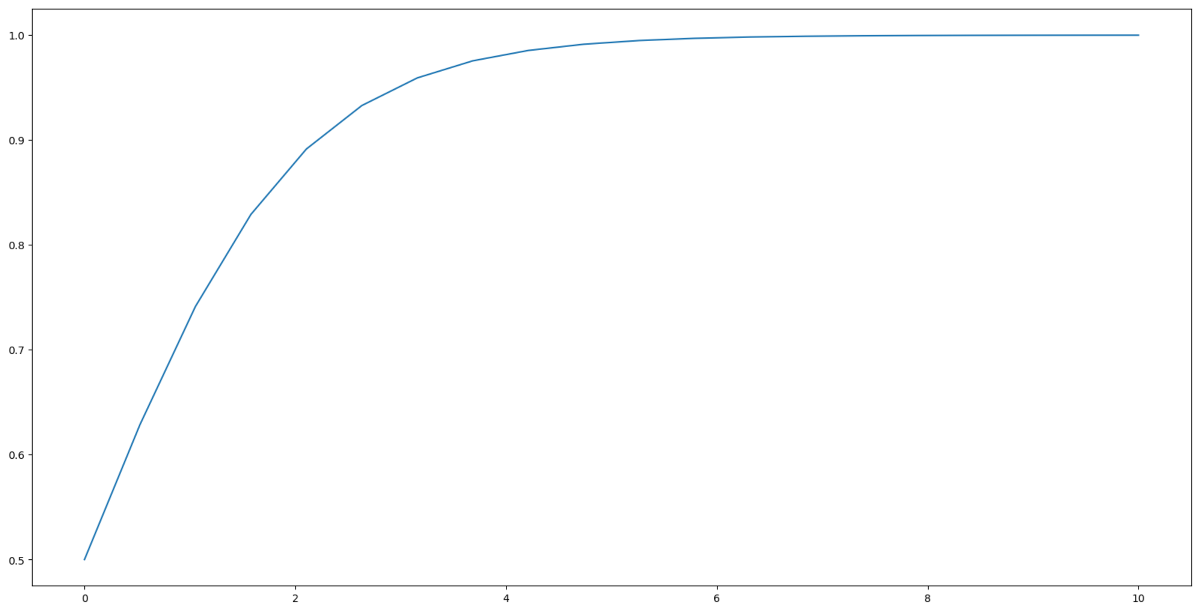
In [7]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1))[:,1])
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x7ff19a874370>]
```



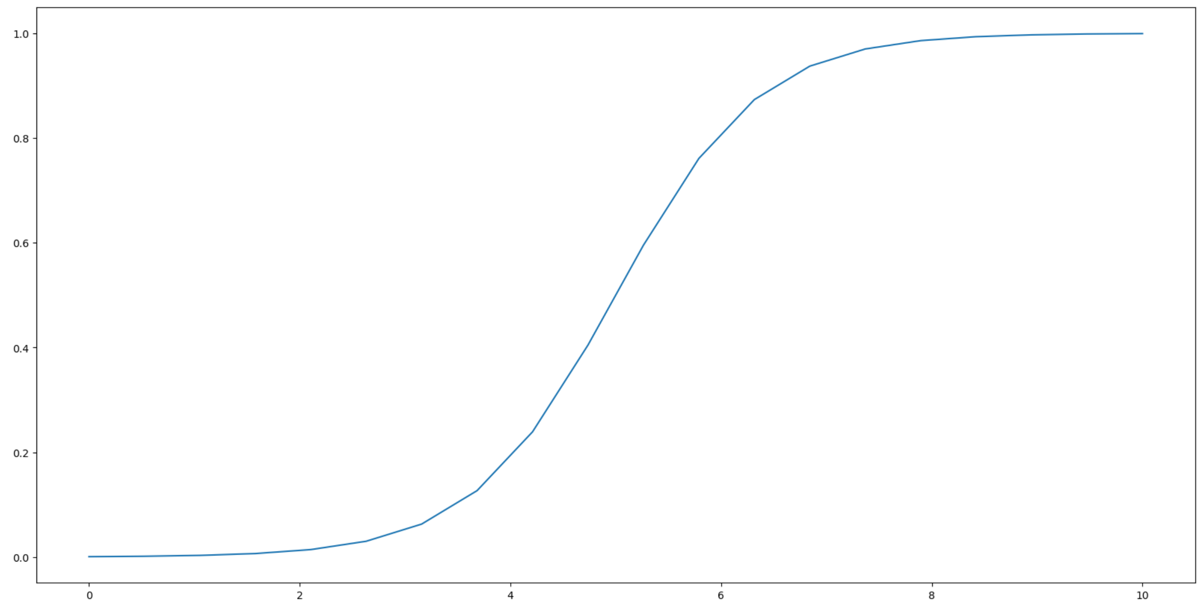
In [8]:
```
b, b0 = model.coef_, model.intercept_
model.coef_, model.intercept_
```
Out[8]:
```
(array([[1.46709085]]), array([-7.33542562]))
```

In [9]:
```
plt.plot(x, 1/(1+np.exp(-x)))
```
Out[9]:
```
[<matplotlib.lines.Line2D at 0x7ff19ae20820>]
```



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 0x7ff19b38ac70>]
```



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')

# 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: 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 [14]:
```
Y
```

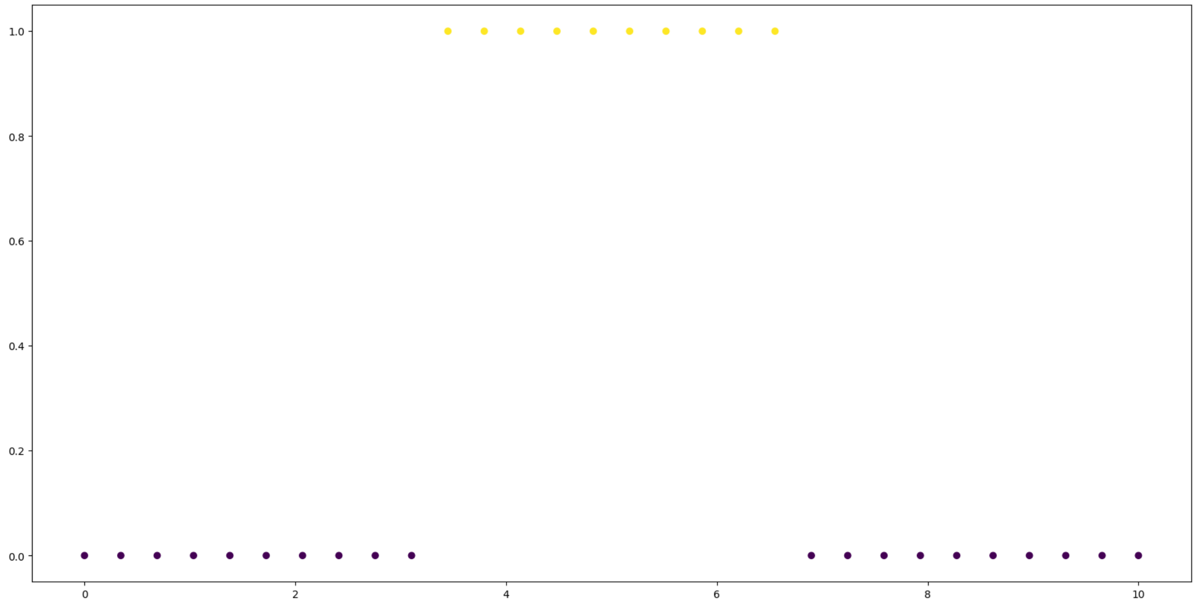
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 Y

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

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

In [15]:
```
y = np.concatenate([np.zeros(10), np.ones(10), np.zeros(10)])
x = np.linspace(0, 10, len(y))
```

In [16]:
```
plt.scatter(x,y, c=y)
```

Out[16]: <matplotlib.collections.PathCollection at 0x7ff19b9b0940>

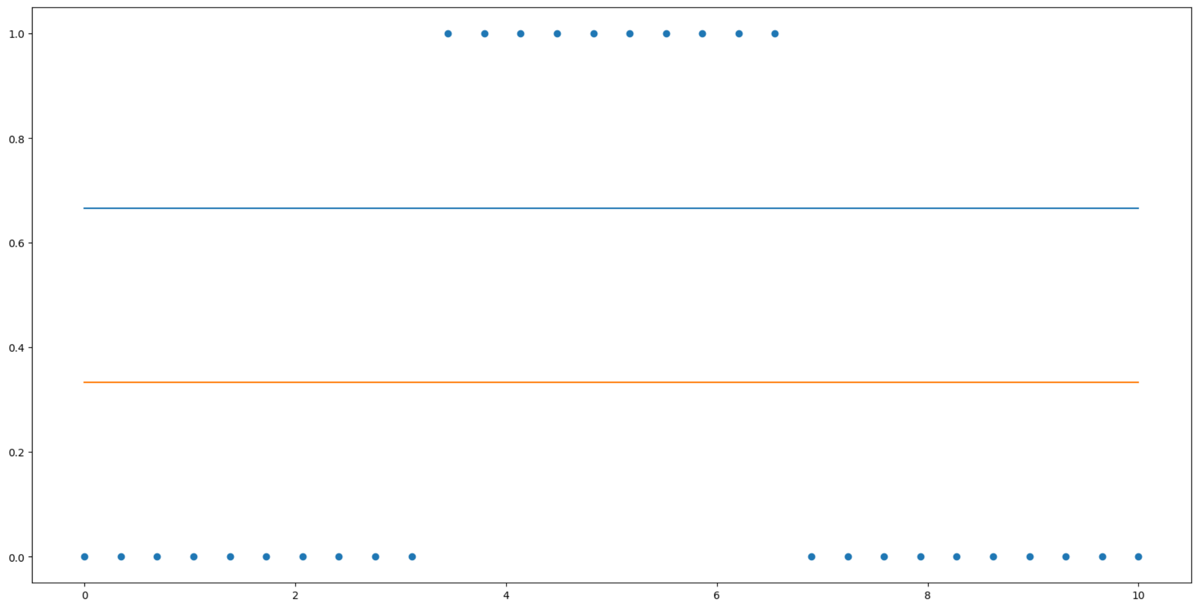


In [17]:
```
model.fit(x.reshape(-1, 1),y)
```

Out[17]:
```
▾ LogisticRegression
LogisticRegression()
```

In [18]:
```
plt.scatter(x,y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1)))
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x7ff17c91a880>,
 <matplotlib.lines.Line2D at 0x7ff17c91a8e0>]
```



In [19]:
```
model1 = LogisticRegression()
model1.fit(x[:15].reshape(-1, 1),y[:15])
```

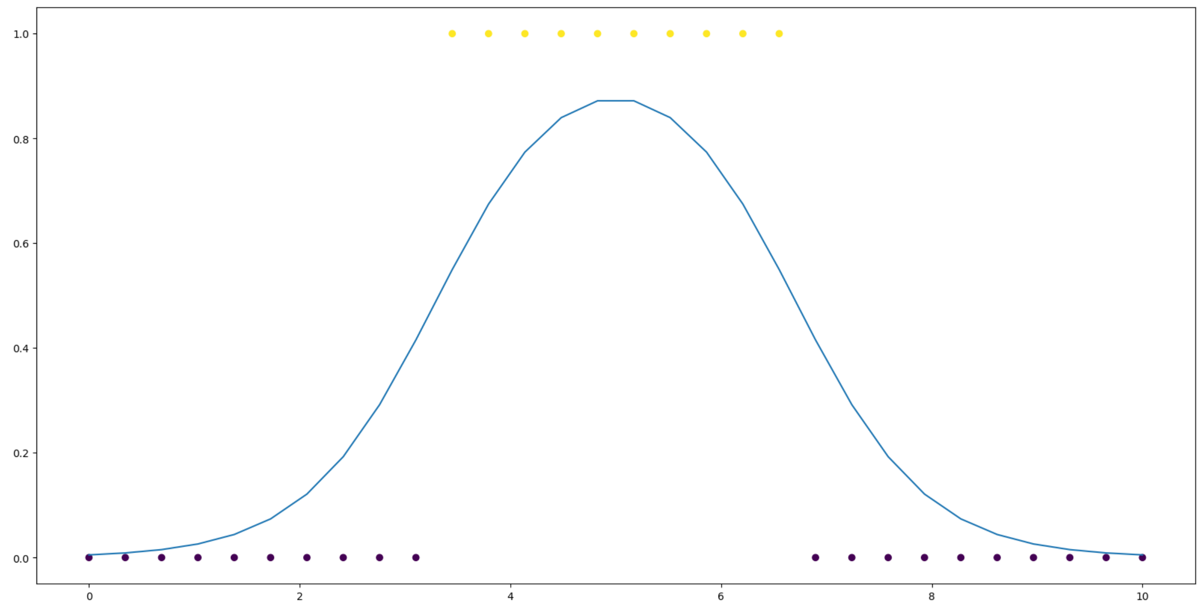
Out[19]:
```
▾ LogisticRegression
LogisticRegression()
```

In [20]:
```
model2 = LogisticRegression()
model2.fit(x[15:].reshape(-1, 1),y[15:])
```

Out[20]:
```
▾ LogisticRegression
LogisticRegression()
```

In [21]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model1.predict_proba(x.reshape(-1, 1))[:,1] * model2.predict_pro
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x7ff17c986430>]
```



In [22]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [23]:
```
from sklearn import preprocessing

enc = preprocessing.OrdinalEncoder()
```

In [24]:
```
transform_columns = ['sex', 'workclass', 'education', 'marital-status',
                     'occupation', 'relationship', 'race', 'sex',
                     'native-country', 'salary']
```

In [25]:
```
x = df.copy()

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

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

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

In [26]:
```
df.salary.unique()
```

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

In [27]:
```
golden.salary.replace(' <=50K.', ' <=50K').replace(' >50K.', ' >50K').unique
```

Out[27]:
```
array([' <=50K', ' >50K'], dtype=object)
```

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

Out[28]:
```
▾ LogisticRegression
LogisticRegression()
```

In [29]:
```
pred = model.predict(preprocessing.scale(x.drop('salary', axis=1)))
pred_test = model.predict(preprocessing.scale(xt.drop('salary', axis=1)))
```

In [30]:
```
x.head()
```

Out[30]:

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

In [31]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

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

Out[32]:
```
0.8250360861152913
```

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

Out[33]:
```
array([[23300,  1420],
       [ 4277,  3564]])
```

In [34]:
```
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 [35]:
```
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

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

1A - Logistic Regression Model

In [69]:
```
# Read in the CSV
heart = pd.read_csv('Heart.csv')
heart.head()
```

Out[69]:

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

In [70]:
```
# Check out the Datatypes
heart.dtypes
```

Out[70]:
```
Unnamed: 0      int64
Age             int64
Sex             int64
ChestPain      object
RestBP          int64
Chol            int64
Fbs             int64
RestECG         int64
MaxHR           int64
ExAng           int64
Oldpeak       float64
Slope           int64
Ca            float64
Thal           object
AHD            object
dtype: object
```

In [71]:
```
# Do preprocessing
from sklearn import preprocessing
enc = preprocessing.OrdinalEncoder()
heart.dropna(inplace=True)
```

In [72]:
```
# Do preprocessing
transform_columns = ['ChestPain', 'Thal', 'AHD']
```

In [73]:
```
# Do x, transform string objects
hdf = heart.copy()
hdf[transform_columns] = enc.fit_transform(heart[transform_columns])
hdf
```

Out[73]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpea
0	1	63	1	3.0	145	233	1	2	150	0	2.
1	2	67	1	0.0	160	286	0	2	108	1	1.
2	3	67	1	0.0	120	229	0	2	129	1	2.
3	4	37	1	1.0	130	250	0	0	187	0	3.
4	5	41	0	2.0	130	204	0	2	172	0	1.
...	...	...	...	...	...	...	...	...	...	...	.
297	298	57	0	0.0	140	241	0	0	123	1	0.
298	299	45	1	3.0	110	264	0	0	132	0	1.
299	300	68	1	0.0	144	193	1	0	141	0	3.
300	301	57	1	0.0	130	131	0	0	115	1	1.
301	302	57	0	2.0	130	236	0	2	174	0	0.

297 rows × 15 columns

In [74]:
```
# Split data into features (X) and target variable (y)
x = hdf.drop('AHD', axis=1)
y = hdf['AHD']
```

In [75]:
```
# Split data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, rar
```

In [76]:
```
# Bring in Logistic Regression model
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

In [77]:
```
# Make predictions
pred = model.predict(preprocessing.scale(x_train))
pred_test = model.predict(preprocessing.scale(x_test))
```

In [78]:
```
# Get ready for reports
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [79]:
```
# Accuracy Score
accuracy_score(y_test, pred_test)
```

Out[79]:
```
0.7444444444444445
```

In [80]:
```
# Confusion Matrix
confusion_matrix(y_test, pred_test)
```

Out[80]:
```
array([[48,  1],
       [22, 19]])
```

In [81]:
```
# Classification Report
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

         0.0       0.69      0.98      0.81        49
         1.0       0.95      0.46      0.62        41

    accuracy                           0.74        90
   macro avg       0.82      0.72      0.71        90
weighted avg       0.81      0.74      0.72        90
```

1B - Shallow Decision Tree Model

In [82]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(hdf.drop(columns=['AHD']

# Create a decision tree classifier with max_depth = 3
clf = DecisionTreeClassifier(max_depth=3)

# Fit the classifier on the training data
clf.fit(X_train, y_train)

# Use the classifier to predict the labels of the test data
y_pred = clf.predict(X_test)

# Print the classification report and confusion matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.74      0.82      0.78        49
         1.0       0.75      0.66      0.70        41

    accuracy                           0.74        90
   macro avg       0.75      0.74      0.74        90
weighted avg       0.74      0.74      0.74        90

[[40  9]
 [14 27]]
```

1C - Optimal Model Selection

The accuracy of both models is the same, at 74%. In fact, both models have similar accuracy, precision, recall, and f1-score. The Logistic Regression Model has a higher precision for class 1.0 (AHD) but lower recall, while the shallow decision tree has a slightly higher recall for class 1.0, but lower precision. Based on this, I would select the Logistic Regresion Model.

2 - Overfit Decision Tree Model

In [83]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(hdf.drop(columns=['AHD']

# Create a decision tree classifier with max_depth = 10
clf = DecisionTreeClassifier(max_depth=10)

# Fit the classifier on the training data
clf.fit(X_train, y_train)

# Use the classifier to predict the labels of the test data
y_pred = clf.predict(X_test)

# Print the classification report and confusion matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.71      0.76      0.73        49
         1.0       0.68      0.63      0.66        41

    accuracy                           0.70        90
   macro avg       0.70      0.69      0.70        90
weighted avg       0.70      0.70      0.70        90

[[37 12]
 [15 26]]
```

The deep decision Tree is the worst of all models tested in terms of accuracy, precision, recall, and F1. It also has reduced computational efficency. By increasing the maximum depth of the decision tree to 10, we are allowing the tree to become much more complex and potentially overfit to the training data. This would have resulted in higher accuracy on the training data, but potentially worse performance on new, unseen data.

In [ ]: