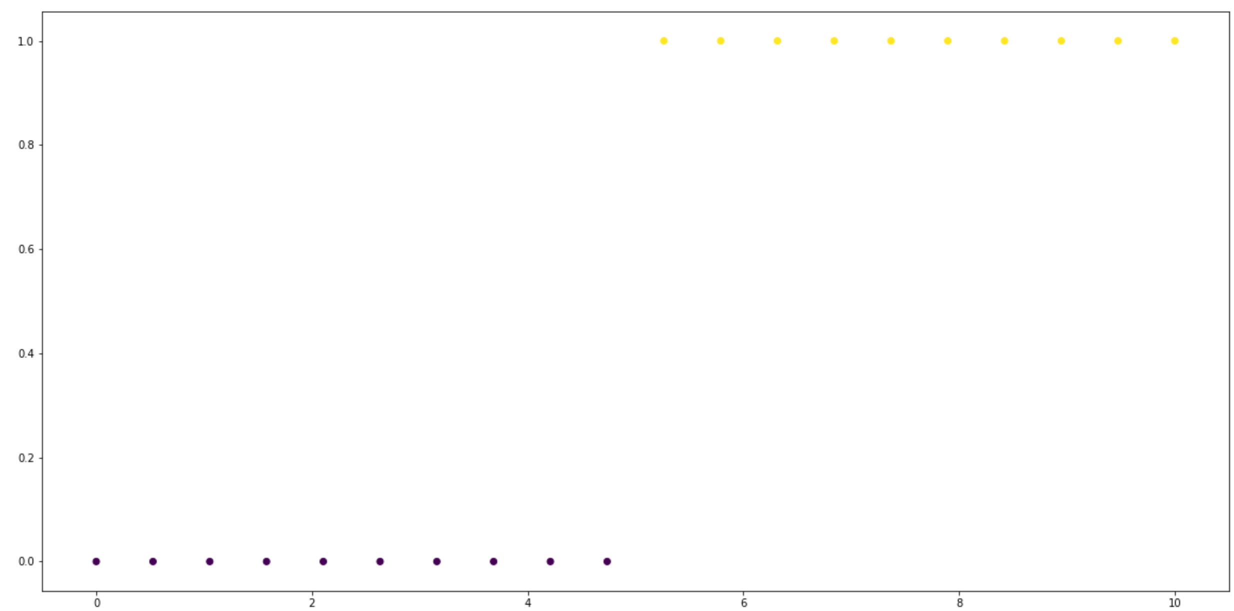
Assignment is at the bottom!

In [1]:
```python
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 [2]:
```python
# creating dummy dataset with 10 datapoints of 0s and 1s to set up classification
# will treat 0s and 1s as probabilities

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

In [3]:
```python
plt.scatter(x, y, c=y)
```

Out[3]:
```
<matplotlib.collections.PathCollection at 0x1e38dd46308>
```



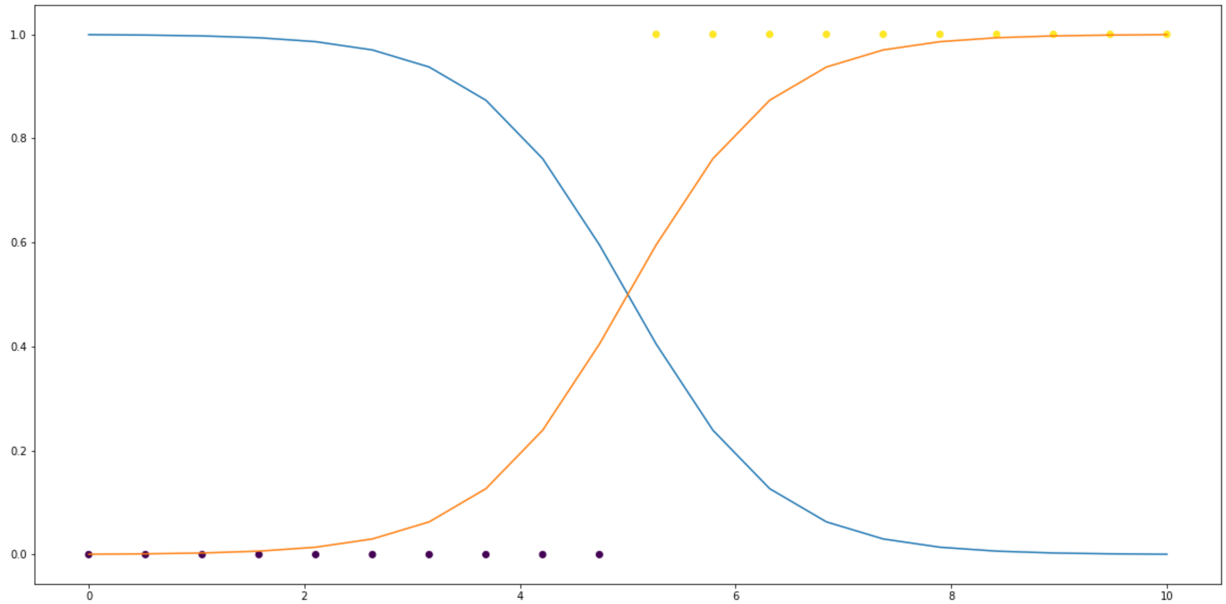
In [4]:
```python
model = LogisticRegression()
```

In [5]:
```python
model.fit(x.reshape(-1, 1),y)
```

Out[5]:
```
LogisticRegression()
```

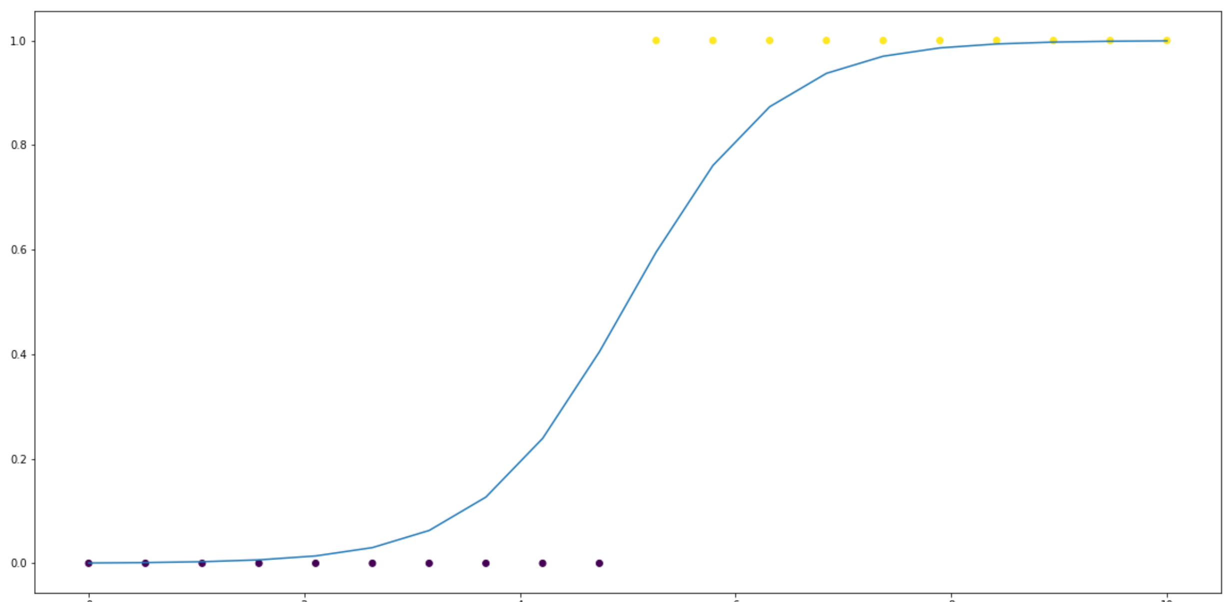
In [6]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1)))
```

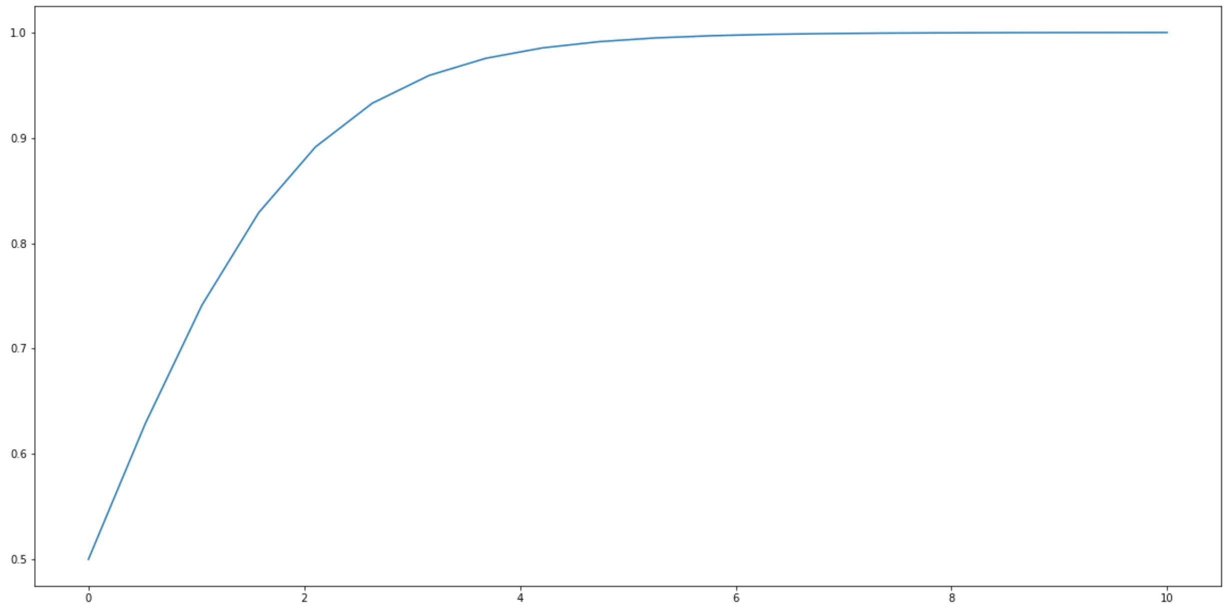
Out[6]:
```
[<matplotlib.lines.Line2D at 0x1e38ddd0f48>,
 <matplotlib.lines.Line2D at 0x1e38e20a708>]
```

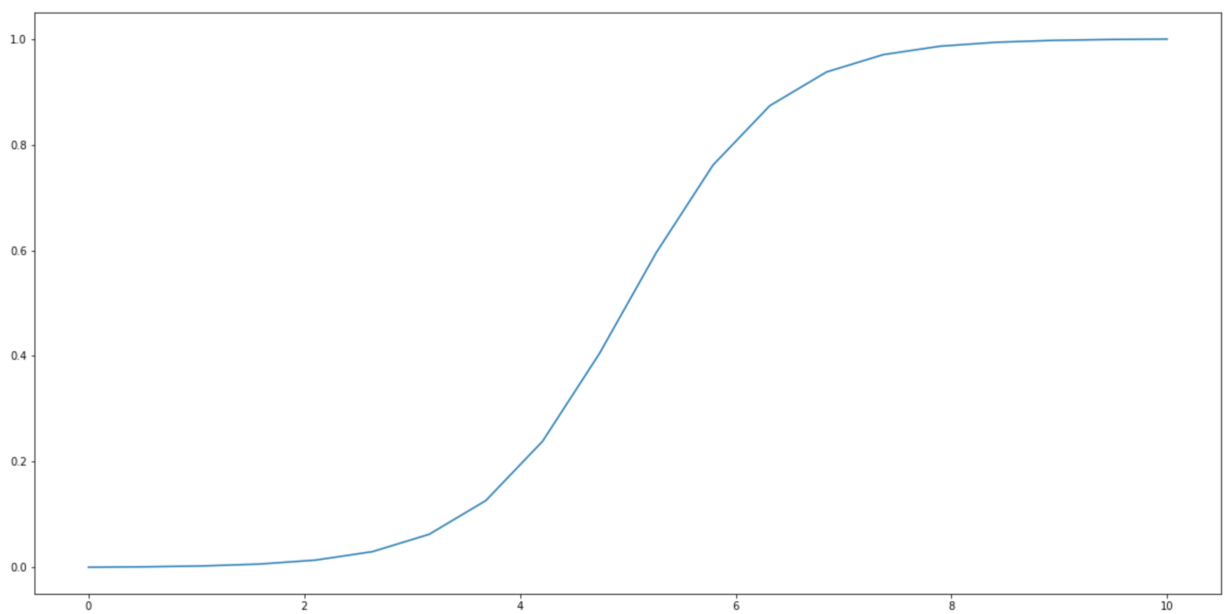


In [7]:
```
# only concerned with probablity of orange line (above)
plt.scatter(x,y, c=y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1))[:,1])
```

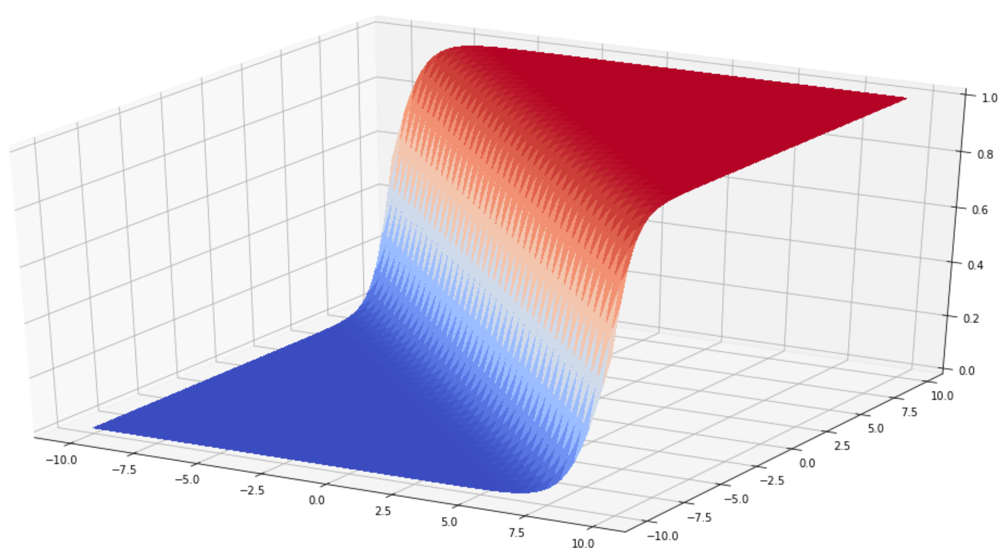
Out[7]:
```
[<matplotlib.lines.Line2D at 0x1e38de0c588>]
```



In [8]:
```
b, b0 = model.coef_, model.intercept_
model.coef_, model.intercept_
```
Out[8]:
```
(array([[1.46709085]]), array([-7.33542562]))
```
In [9]:
```
# shows only half of the signoid
plt.plot(x, 1/(1+np.exp(-x)))
```
Out[9]:
```
[<matplotlib.lines.Line2D at 0x1e38e151108>]
```


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 0x1e38e745808>]
```


In [12]:
```
# 8:50 of lecture

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) # these are matrixes
R = np.sqrt(X**2 + Y**2)
# Z = 1/(1+np.exp(-(b[0]*X +b[0]*Y +b0))) # less steep
# Z = 1/(1+np.exp(-(b[0]*X +.25*b[0]*Y +b0))) # more steep
Z = 1/(1+np.exp(-(b[0]*X +b[0]*Y +.2*b0+2))) #moves the intercept
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
```



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

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

In [14]:
```
Y
```

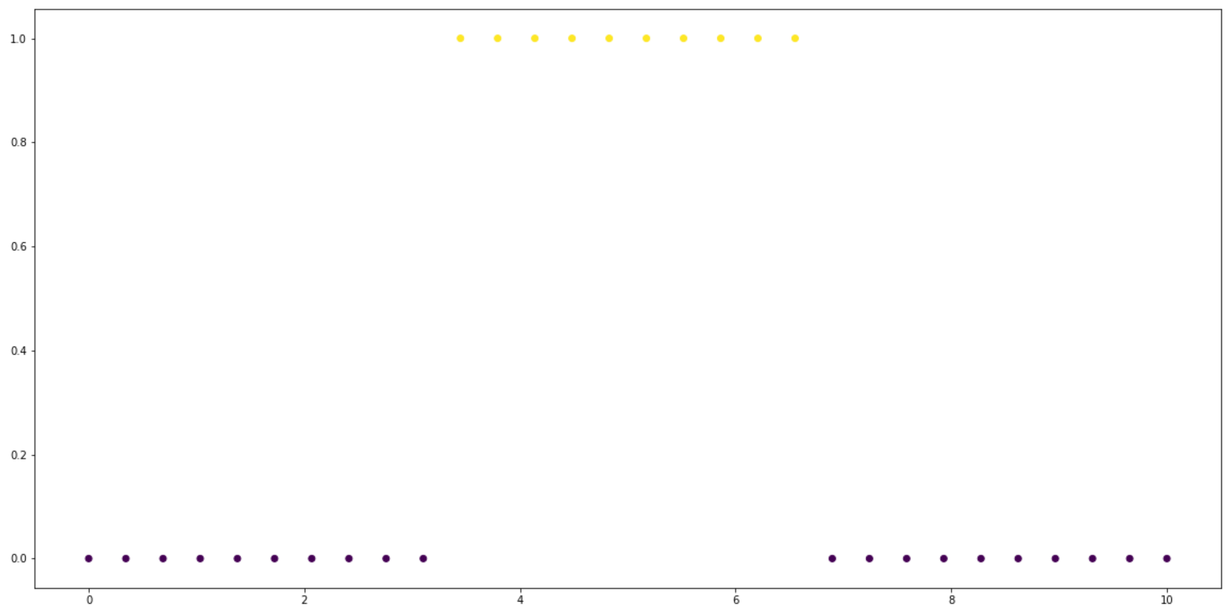
Out[14]:
```
array([[-10.  , -10.  , -10.  , ..., -10.  , -10.  , -10.  ],
       [ -9.75,  -9.75,  -9.75, ...,  -9.75,  -9.75,  -9.75],
       [ -9.5 ,  -9.5 ,  -9.5 , ...,  -9.5 ,  -9.5 ,  -9.5 ],
       ...,
       [  9.25,   9.25,   9.25, ...,   9.25,   9.25,   9.25],
       [  9.5 ,   9.5 ,   9.5 , ...,   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 [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 0x1e38ed43fc8>
```

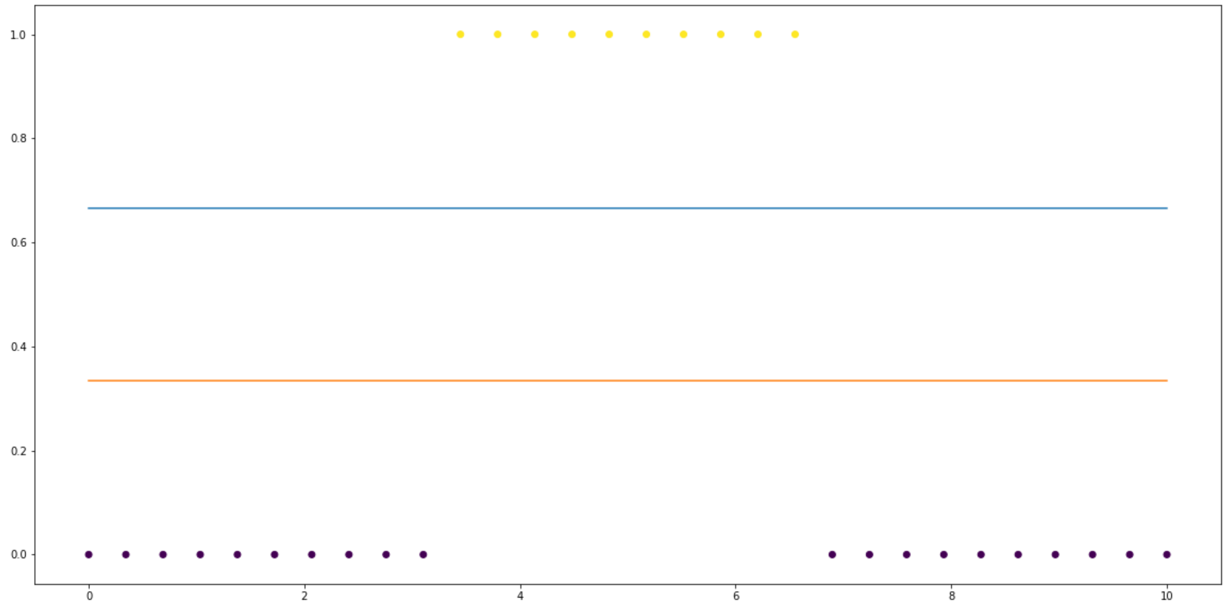


In [17]:
```
model.fit(x.reshape(-1, 1),y)
```

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

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

```
Out[18]: [<matplotlib.lines.Line2D at 0x1e38e870308>,
          <matplotlib.lines.Line2D at 0x1e38ecd0c88>]
```




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

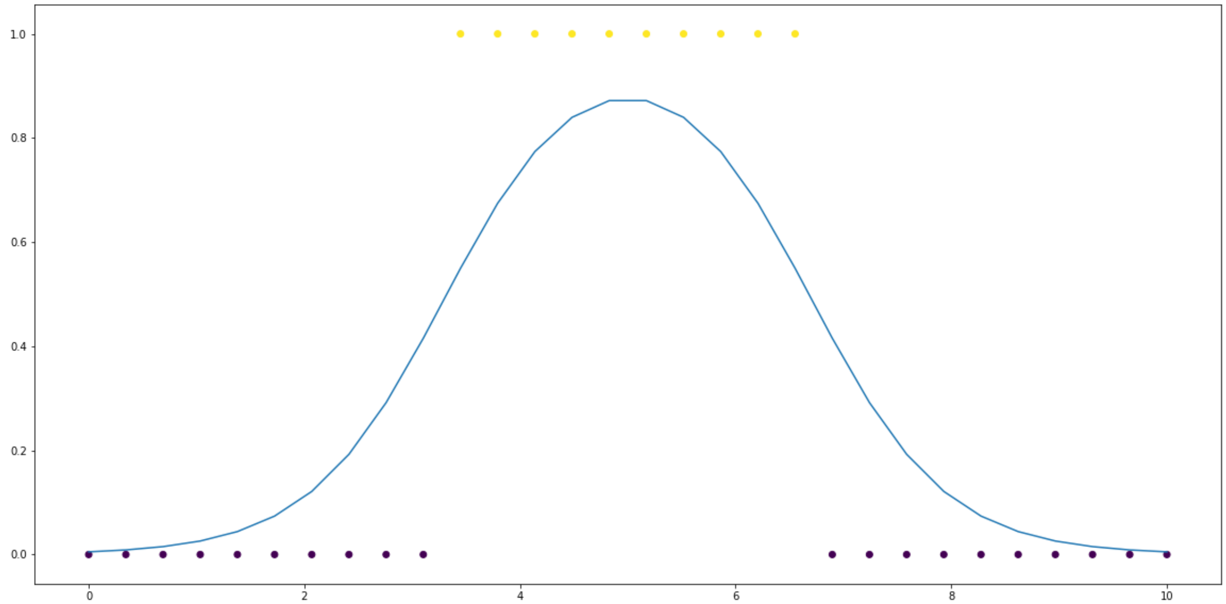
```
Out[19]: LogisticRegression()
```

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

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

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

Out[21]:
```
[<matplotlib.lines.Line2D at 0x1e38ecfab88>]
```



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]:
```python
df.salary.unique()
```
Out[26]: array([' <=50K', ' >50K'], dtype=object)

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

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

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

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

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

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

In [32]:
```python
accuracy_score(x.salary, pred)
```
Out[32]: 0.8250360861152913

In [33]:
```python
confusion_matrix(x.salary, pred)
```
Out[33]:
```
array([[23300,  1420],
       [ 4277,  3564]], dtype=int64)
```

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 (create a train and a test set) and build 2 models: Logistic Regression and Decision Tree (shallow (2-3)). Compare the test results.

2. Repeat 1. but let the Decision Tree be much deeper to allow over-fitting. Compare the two models' test results again, does the Logistic Regression have an improvement due to a lower variance?

In [36]:
```
insurance = pd.read_csv('../data/insurance.csv')
insurance.head()
```

Out[36]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [37]:
```
insurance.dtypes
```

Out[37]:
```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

In [38]:
```
from sklearn import preprocessing
enc = preprocessing.OrdinalEncoder()
```

In [39]:
```
transform_columns = ['sex', 'smoker', 'region']
```

In [40]:
```
x = insurance.copy()
x[transform_columns] = enc.fit_transform(insurance[transform_columns])
x
```

Out[40]:

	age	sex	bmi	children	smoker	region	charges
0	19	0.0	27.900	0	1.0	3.0	16884.92400
1	18	1.0	33.770	1	0.0	2.0	1725.55230
2	28	1.0	33.000	3	0.0	2.0	4449.46200
3	33	1.0	22.705	0	0.0	1.0	21984.47061
4	32	1.0	28.880	0	0.0	1.0	3866.85520
...	...	...	...	...	...	...	...
1333	50	1.0	30.970	3	0.0	1.0	10600.54830
1334	18	0.0	31.920	0	0.0	0.0	2205.98080
1335	18	0.0	36.850	0	0.0	2.0	1629.83350
1336	21	0.0	25.800	0	0.0	3.0	2007.94500
1337	61	0.0	29.070	0	1.0	1.0	29141.36030

1338 rows × 7 columns

In [41]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x.drop('smoker', axis=1), x.s
```

In [42]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [43]:
```
model.fit(preprocessing.scale(x_train), y_train)
model.coef_, model.intercept_
```

Out[43]:
```
(array([[-1.07041457,  0.10937251, -1.66413989, -0.22858889,  0.08491027,
          3.89658144]]),
 array([-3.44537705]))
```

In [44]:
```
pred = model.predict(preprocessing.scale(x_train))
pred_test = model.predict(preprocessing.scale(x_test))
```

In [45]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [46]:
```
accuracy_score(y_train, pred)
```

Out[46]:
```
0.9601196410767697
```

In [47]:
```
accuracy_score(y_test, pred_test)
```

Out[47]:
```
0.9522388059701492
```

In [48]:
```
confusion_matrix(y_test, pred_test)
```

Out[48]:
```
array([[258,   9],
       [  7,  61]], dtype=int64)
```

In [49]:
```
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

         0.0       0.97      0.97      0.97       267
         1.0       0.87      0.90      0.88        68

    accuracy                           0.95       335
   macro avg       0.92      0.93      0.93       335
weighted avg       0.95      0.95      0.95       335
```

Decision Tree at 2 levels

In [50]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [51]:
```
model = DecisionTreeClassifier(criterion='entropy', max_depth=2)
model.fit(x_train, y_train)
```

Out[51]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

In [52]:
```
model.tree_.node_count
```

Out[52]:
```
7
```

In [53]:
```
pred = model.predict(x_train)
pred_test = model.predict(x_test)
```

In [54]:
```
accuracy_score(y_train, pred)
```
Out[54]:
```
0.9292123629112662
```

In [55]:
```
accuracy_score(y_test, pred_test)
```
Out[55]:
```
0.9104477611940298
```

In [56]:
```
print(classification_report(y_test, pred_test))
```
```
              precision    recall  f1-score   support

         0.0       1.00      0.89      0.94       267
         1.0       0.69      1.00      0.82        68

    accuracy                           0.91       335
   macro avg       0.85      0.94      0.88       335
weighted avg       0.94      0.91      0.92       335
```

Comparison

The logistic model is more accurate, however, the two level decision tree successful predicts true positives and true negatives but poorly predicts false positives.

2. Repeat 1. but let the Decision Tree be much deeper to allow over-fitting. Compare the two models' test results again, does the Logistic Regression have an improvement due to a lower variance?

In [57]:
```
model = DecisionTreeClassifier(criterion='entropy', max_depth=10)
model.fit(x_train, y_train)
```
Out[57]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=10)
```

In [58]:
```
model.tree_.node_count
```
Out[58]:
```
61
```

In [59]:
```
pred = model.predict(x_train)
pred_test = model.predict(x_test)
```

In [60]:
```
accuracy_score(y_train, pred)
```
Out[60]:
```
0.9920239282153539
```

In [61]:
```
accuracy_score(y_test, pred_test)
```
Out[61]:
```
0.9582089552238806
```

In [62]:
```
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

         0.0       0.97      0.98      0.97       267
         1.0       0.92      0.87      0.89        68

    accuracy                           0.96       335
   macro avg       0.94      0.92      0.93       335
weighted avg       0.96      0.96      0.96       335
```

Comparison

The 10 level decision tree performs better that prior logistic and two level decision tree, with higher accuracy, precision, and recall. However, the deeper model likely over-fits and may not perform well on new data.