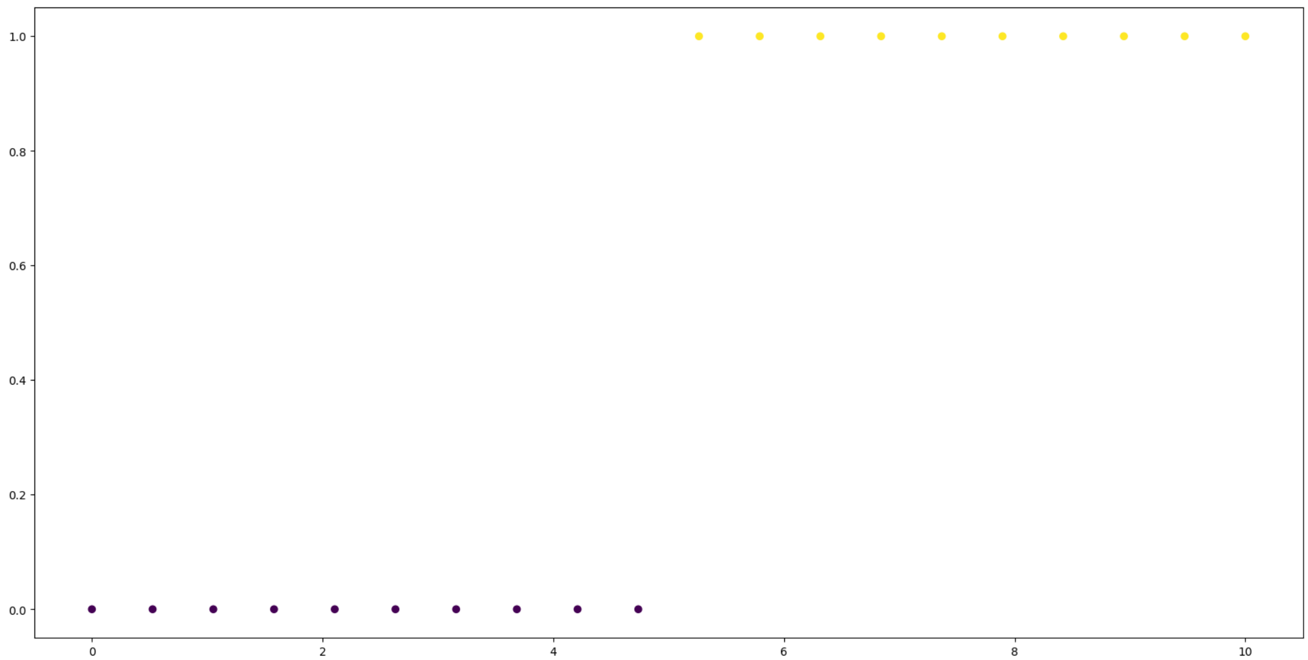
Assignment is at the bottom!

In [44]:
```
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 [45]:
```
y = np.concatenate([np.zeros(10), np.ones(10)])
x = np.linspace(0, 10, len(y))
```

In [46]:
```
plt.scatter(x, y, c=y)
```

Out[46]:
```
<matplotlib.collections.PathCollection at 0x1ab32699280>
```



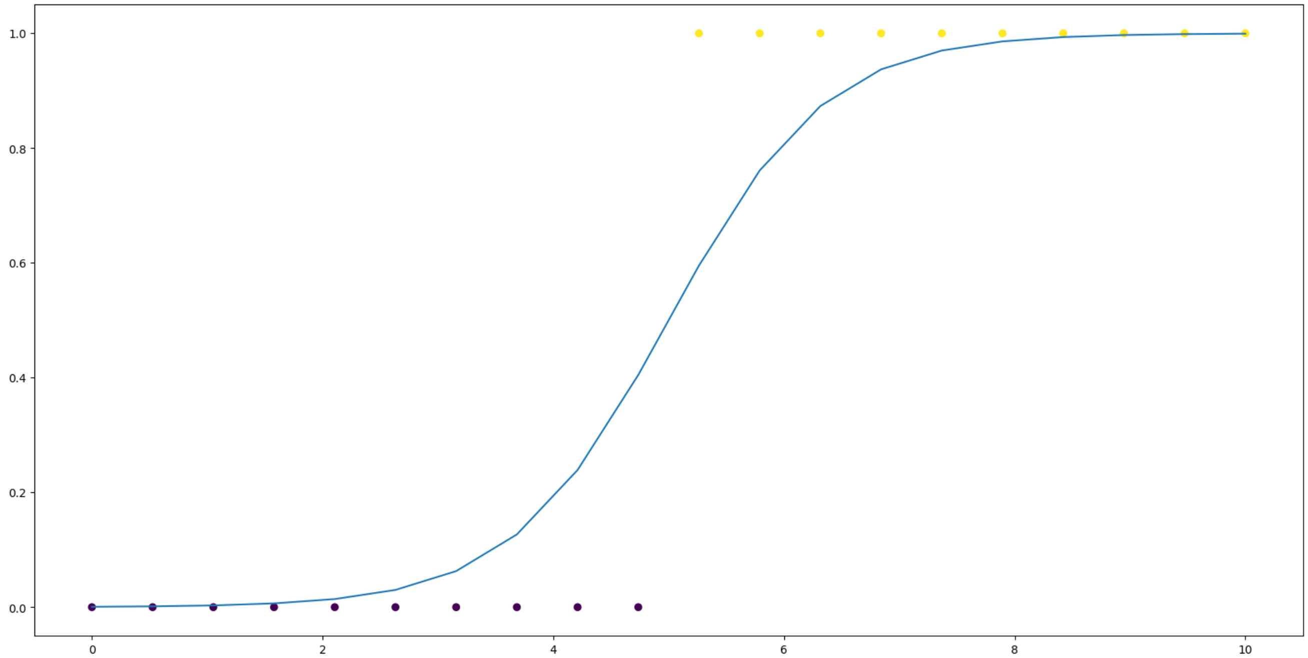
In [47]:
```
model = LogisticRegression()
```

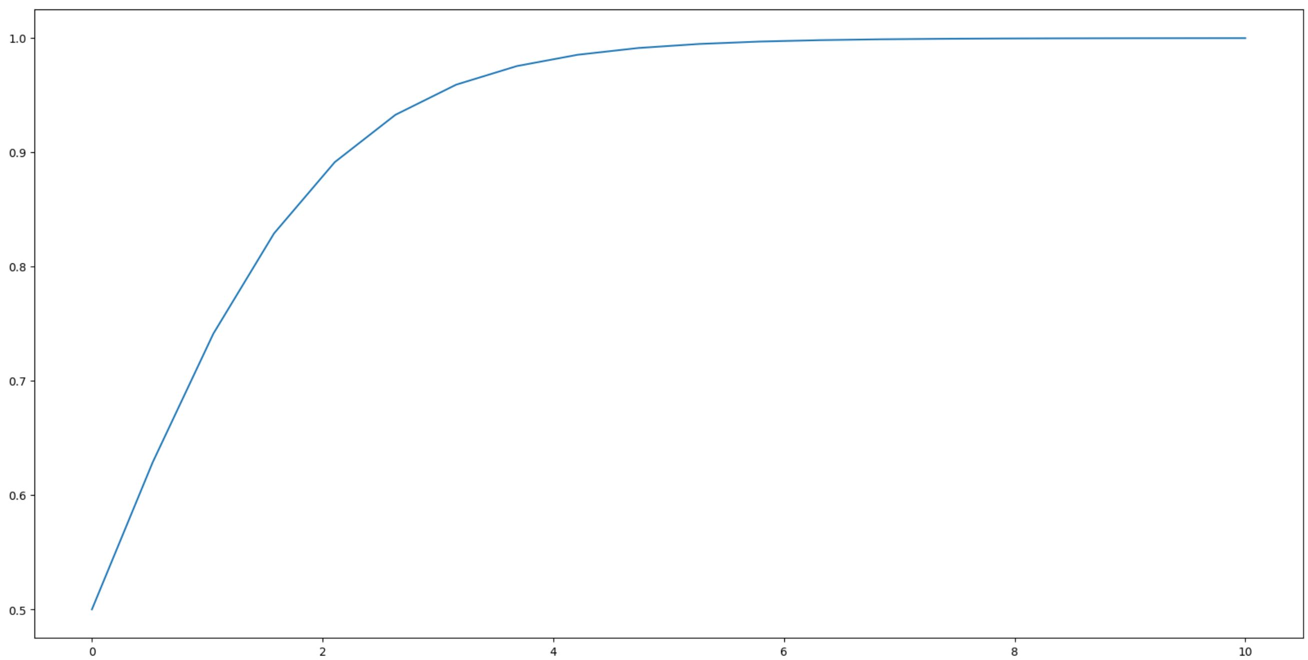
In [48]:
```
model.fit(x.reshape(-1, 1),y)
```

Out[48]:
```
▾ LogisticRegression
LogisticRegression()
```

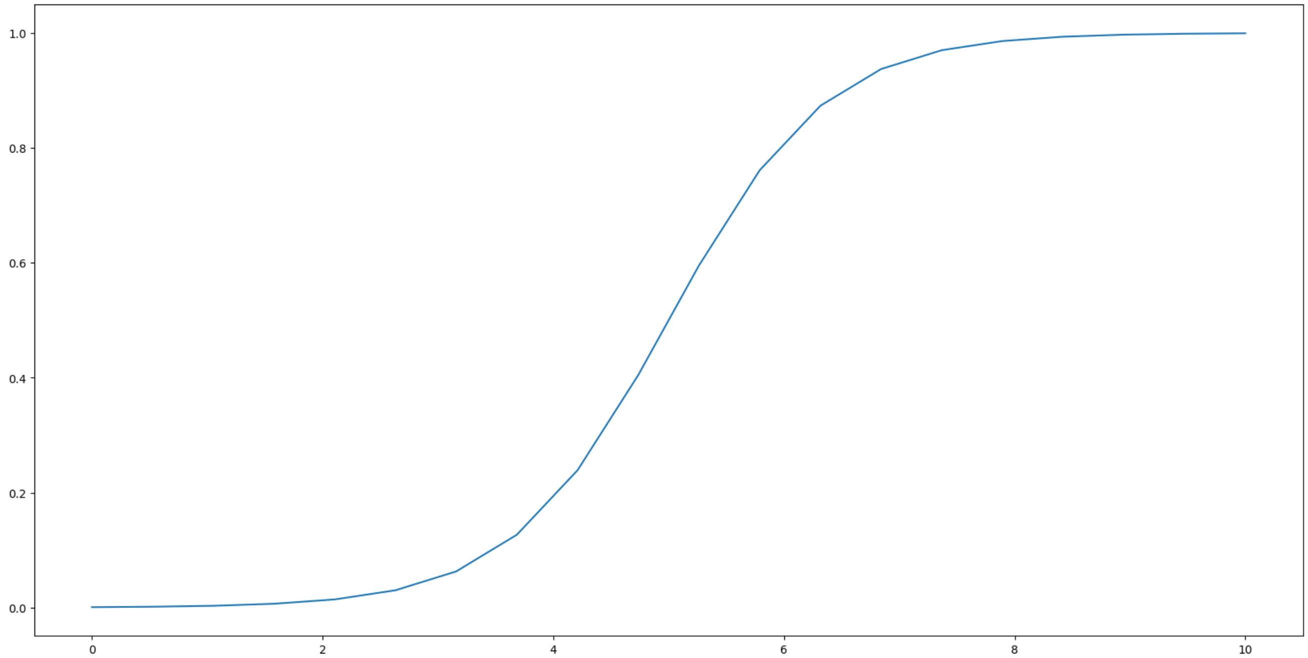
In [49]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1))[:,1])
```

Out[49]:
```
[<matplotlib.lines.Line2D at 0x1ab32bd7a60>]
```

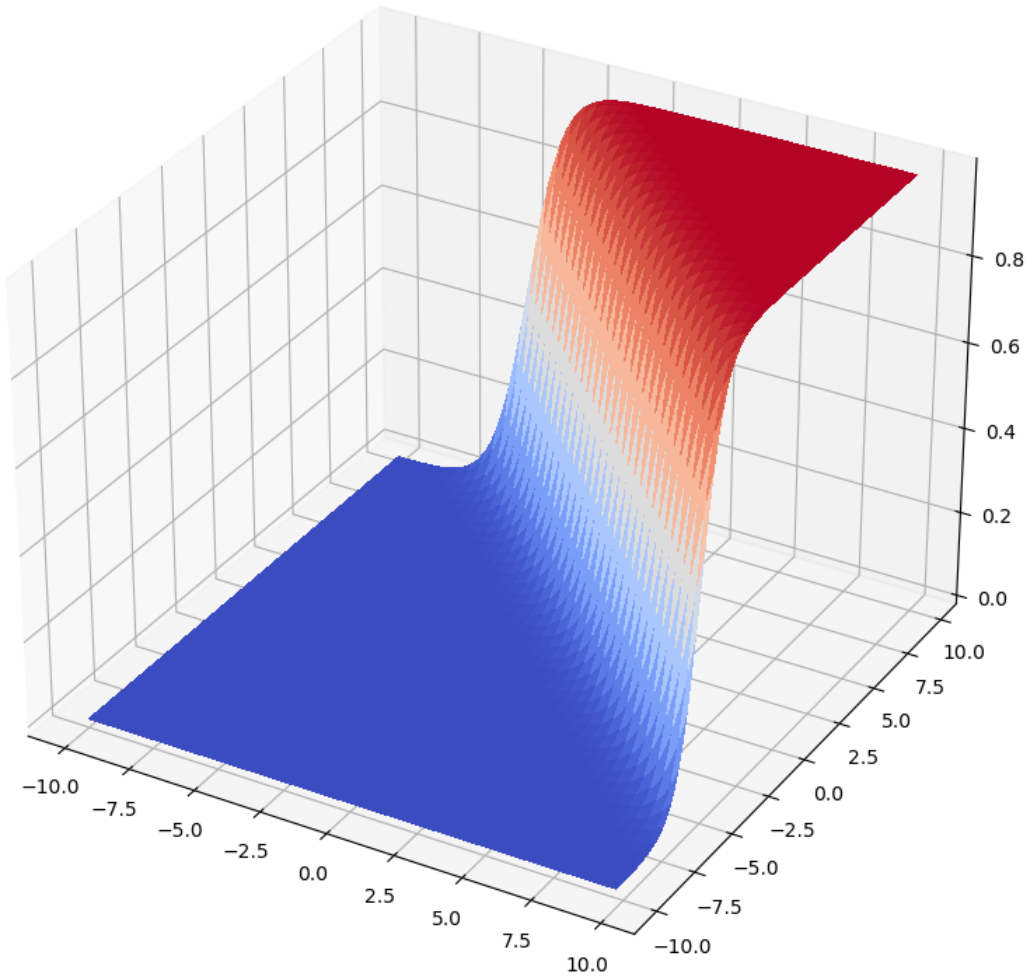


In [50]:
```
b, b0 = model.coef_, model.intercept_
model.coef_, model.intercept_
```
Out[50]:
```
(array([[1.46709085]]), array([-7.33542562]))
```
In [51]:
```
plt.plot(x, 1/(1+np.exp(-x)))
```
Out[51]:
```
[<matplotlib.lines.Line2D at 0x1ab328555b0>]
```


In [52]:
```
b
```
Out[52]:
```
array([[1.46709085]])
```
In [53]:
```
plt.plot(x, 1/(1+np.exp(-(b[0]*x +b0))))
```
Out[53]:
```
[<matplotlib.lines.Line2D at 0x1ab3268fb50>]
```



In [54]:
```
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 = plt.axes(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 [55]:
```
X
```

Out[55]:
```
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 [56]:
```
Y
```

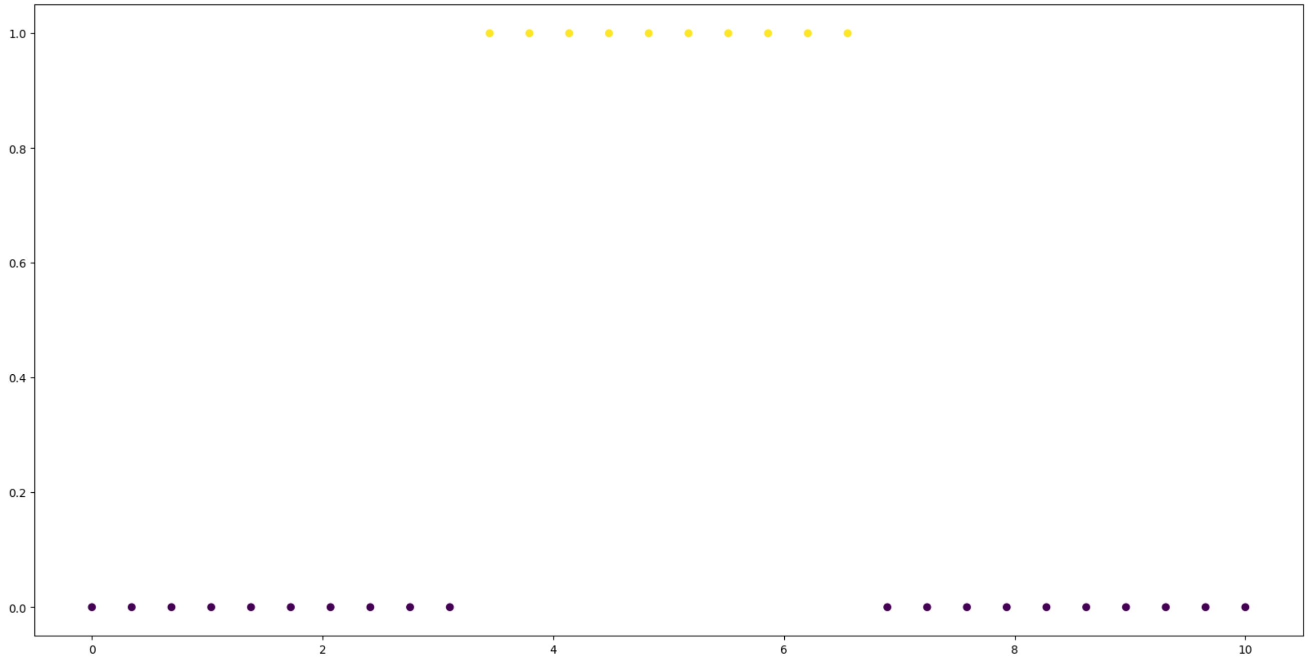
Out[56]:
```
array([[-10.  , -10.  , -10.  , ..., -10.  , -10.  , -10.  ],
       [ -9.75,  -9.75,  -9.75, ...,  -9.75,  -9.75,  -9.75],
       [ -9.5 ,  -9.5 ,  -9.5 , ...,  -9.5 ,  -9.5 ,  -9.5 ],
       ...,
       [  9.25,   9.25,   9.25, ...,   9.25,   9.25,   9.25],
       [  9.5 ,   9.5 ,   9.5 , ...,   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 [57]:
```
y = np.concatenate([np.zeros(10), np.ones(10), np.zeros(10)])
x = np.linspace(0, 10, len(y))
```

In [58]:
```
plt.scatter(x,y, c=y)
```

Out[58]:
```
<matplotlib.collections.PathCollection at 0x1ab32ea84f0>
```

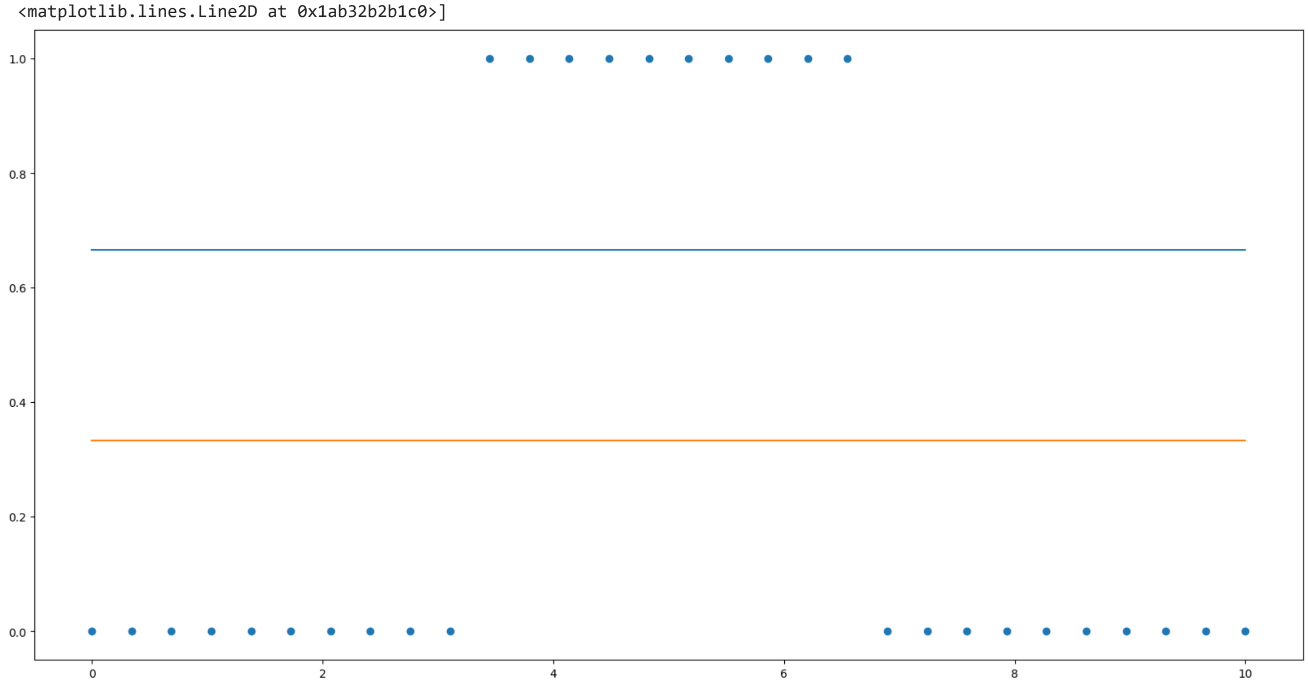


In [59]:
```
model.fit(x.reshape(-1, 1),y)
```

Out[59]:
```
▾ LogisticRegression
LogisticRegression()
```

In [60]:
```
plt.scatter(x,y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1)))
```

Out[60]:
```
[<matplotlib.lines.Line2D at 0x1ab32b2b160>,
 <matplotlib.lines.Line2D at 0x1ab32b2b1c0>]
```



In [61]:
```
model1 = LogisticRegression()
model1.fit(x[:15].reshape(-1, 1),y[:15])
```

Out[61]:
```
▾ LogisticRegression
LogisticRegression()
```

In [62]:
```
model2 = LogisticRegression()
model2.fit(x[15:].reshape(-1, 1),y[15:])
```

Out[62]:

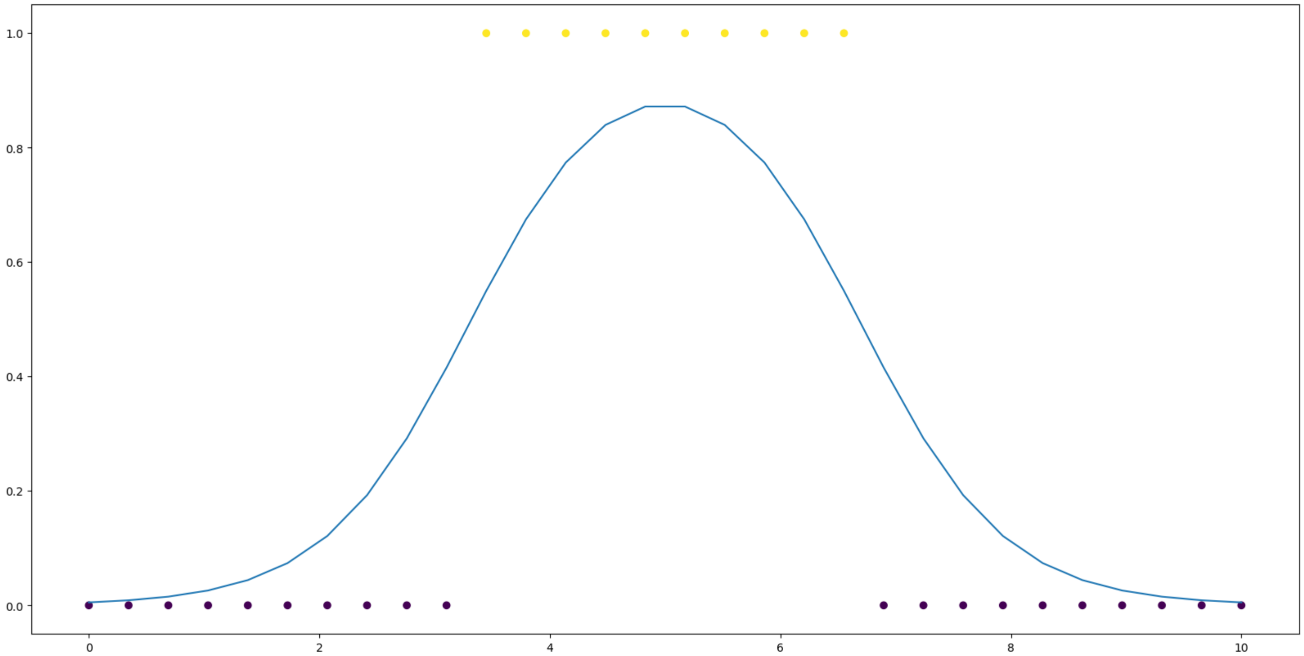
```
▾ LogisticRegression
LogisticRegression()
```

In [63]:

```python
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])
```

Out[63]:

```
[<matplotlib.lines.Line2D at 0x1ab328a9220>]
```



In [109…

```python
df = pd.read_csv('../data/adult.data', index_col=False)
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [110…

```python
from sklearn import preprocessing

enc = preprocessing.OrdinalEncoder()
```

In [111…

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

In [112…

```python
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 [113…

```python
x
```

Out[113]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	7.0	77516	9.0	13	4.0	1.0	1.0	4.0	1.0	2174	0	40	39.0	0.0
1	50	6.0	83311	9.0	13	2.0	4.0	0.0	4.0	1.0	0	0	13	39.0	0.0
2	38	4.0	215646	11.0	9	0.0	6.0	1.0	4.0	1.0	0	0	40	39.0	0.0
3	53	4.0	234721	1.0	7	2.0	6.0	0.0	2.0	1.0	0	0	40	39.0	0.0
4	28	4.0	338409	9.0	13	2.0	10.0	5.0	2.0	0.0	0	0	40	5.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	4.0	257302	7.0	12	2.0	13.0	5.0	4.0	0.0	0	0	38	39.0	0.0
32557	40	4.0	154374	11.0	9	2.0	7.0	0.0	4.0	1.0	0	0	40	39.0	1.0
32558	58	4.0	151910	11.0	9	6.0	1.0	4.0	4.0	0.0	0	0	40	39.0	0.0
32559	22	4.0	201490	11.0	9	4.0	1.0	3.0	4.0	1.0	0	0	20	39.0	0.0
32560	52	5.0	287927	11.0	9	2.0	4.0	5.0	4.0	0.0	15024	0	40	39.0	1.0

32561 rows × 15 columns

In [68]:
```
df.salary.unique()
```
Out[68]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [69]:
```
golden.salary.replace(' <=50K.', ' <=50K').replace(' >50K.', ' >50K').unique()
```
Out[69]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [70]:
```
model.fit(preprocessing.scale(x.drop('salary', axis=1)), x.salary)
```
Out[70]:
```
▾ LogisticRegression
LogisticRegression()
```

In [71]:
```
pred = model.predict(preprocessing.scale(x.drop('salary', axis=1)))
pred_test = model.predict(preprocessing.scale(xt.drop('salary', axis=1)))
```

In [72]:
```
x.head()
```
Out[72]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	7.0	77516	9.0	13	4.0	1.0	1.0	4.0	1.0	2174	0	40	39.0	0.0
1	50	6.0	83311	9.0	13	2.0	4.0	0.0	4.0	1.0	0	0	13	39.0	0.0
2	38	4.0	215646	11.0	9	0.0	6.0	1.0	4.0	1.0	0	0	40	39.0	0.0
3	53	4.0	234721	1.0	7	2.0	6.0	0.0	2.0	1.0	0	0	40	39.0	0.0
4	28	4.0	338409	9.0	13	2.0	10.0	5.0	2.0	0.0	0	0	40	5.0	0.0

In [73]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [74]:
```
accuracy_score(x.salary, pred)
```
Out[74]:
```
0.8250360861152913
```

In [75]:
```
confusion_matrix(x.salary, pred)
```
Out[75]:
```
array([[23300,  1420],
       [ 4277,  3564]], dtype=int64)
```

In [76]:
```
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 [77]:
```
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).

In [123…
```
heart = pd.read_csv('../data/Heart.csv', index_col=False)
heart = heart.copy().drop(["Unnamed: 0"], axis=1)
heart = heart.dropna()

cat_columns = ['ChestPain', 'Thal', 'AHD']
# ChestPain: asymptomatic=0, nonanginal=1, nontypical=2, typical=3
# Thal: fixed=0, normal=1, reversable=2
# ADH: No=0, Yes=1

heart[cat_columns] = enc.fit_transform(heart[cat_columns])

from sklearn.model_selection import train_test_split
hx_train, hx_test, hy_train, hy_test = train_test_split(heart.drop(['AHD'], axis=1),heart.AHD, test_size=.20)
```

In [162…
```
heart.head()
```

Out[162]:

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

In [208…
```
from sklearn.tree import DecisionTreeClassifier
model_tree = DecisionTreeClassifier(criterion='entropy', max_depth=2)
model_tree.fit(hx_train, hy_train)
p_tree = model_tree.predict(hx_test)
```

In [209…
```
model_logit = LogisticRegression()
model_logit.fit(preprocessing.scale(hx_train), hy_train)
p_logit = model_logit.predict(preprocessing.scale(hx_test))
```

Compare the test results using `classification_report` and `confusion_matrix`.

Decision Tree Model

In [210…
```
confusion_matrix(hy_test, p_tree)
```

Out[210]:
```
array([[28,  1],
       [17, 14]], dtype=int64)
```

In [211…
```
print(classification_report(hy_test, p_tree))
```

```
              precision    recall  f1-score   support

         0.0       0.62      0.97      0.76        29
         1.0       0.93      0.45      0.61        31

    accuracy                           0.70        60
   macro avg       0.78      0.71      0.68        60
weighted avg       0.78      0.70      0.68        60
```

Logit Model

In [212…
```
confusion_matrix(hy_test, p_logit)
```

Out[212]:
```
array([[26,  3],
       [ 9, 22]], dtype=int64)
```

In [213…
```
print(classification_report(hy_test, p_logit))
```

```
              precision    recall  f1-score   support

         0.0       0.74      0.90      0.81        29
         1.0       0.88      0.71      0.79        31

    accuracy                           0.80        60
   macro avg       0.81      0.80      0.80        60
weighted avg       0.81      0.80      0.80        60
```

Explain which algorithm is optimal

Compared to a shallow decision tree (max-depth=2), the logistic regression model preforms better overall - higher f1-scores for both categories (in this case, correctly determining which patients have AHD and which do not) and overall accuracy. The logit model does have more false positives than the decision tree model though, meaning that there was a higher percentage of patients determined to be positive for AHD when they weren't.

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 [214…
```
model_tree2 = DecisionTreeClassifier(criterion='entropy', max_depth=10)
model_tree2.fit(hx_train, hy_train)
p_tree2 = model_tree2.predict(hx_test)
```

In [215…
```
confusion_matrix(hy_test, p_tree2)
```

Out[215]:
```
array([[25,  4],
       [13, 18]], dtype=int64)
```

In [216…
```
print(classification_report(hy_test, p_tree2))
```

```
              precision    recall  f1-score   support

         0.0       0.66      0.86      0.75        29
         1.0       0.82      0.58      0.68        31

    accuracy                           0.72        60
   macro avg       0.74      0.72      0.71        60
weighted avg       0.74      0.72      0.71        60
```

In [217…
```
confusion_matrix(hy_test, p_logit)
```

Out[217]:
```
array([[26,  3],
       [ 9, 22]], dtype=int64)
```

In [218…
```
print(classification_report(hy_test, p_logit))
```

```
              precision    recall  f1-score   support

         0.0       0.74      0.90      0.81        29
         1.0       0.88      0.71      0.79        31

    accuracy                           0.80        60
   macro avg       0.81      0.80      0.80        60
weighted avg       0.81      0.80      0.80        60
```

When the decision tree model is allowed to overfit itself to the training data, the accuracy of the model increases slightly overall (fewer false negatives though more false positives), but in just about every dimension the logit model preforms better.