Casey Masamitsu | Week 7 Assignment | MLNN

Assignment is at the bottom!

In [46]:
```
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 [47]:
```
y = np.concatenate([np.zeros(10), np.ones(10)])
x = np.linspace(0, 10, len(y))
```

In [49]:
```
x
```

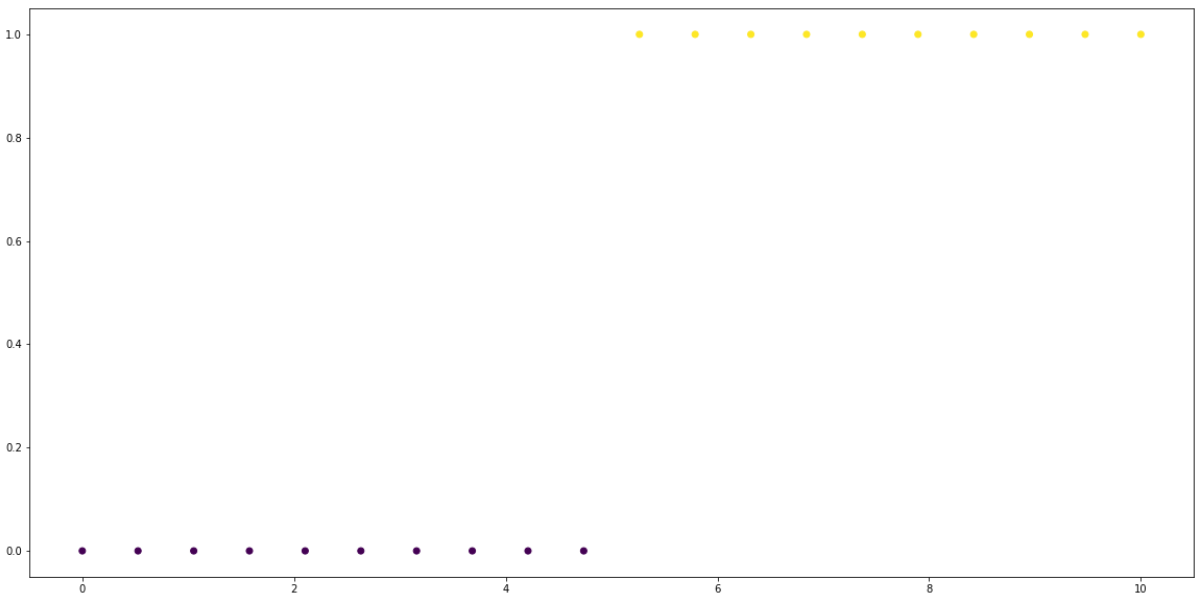
Out[49]:
```
array([ 0.        ,  0.52631579,  1.05263158,  1.57894737,  2.10526316,
        2.63157895,  3.15789474,  3.68421053,  4.21052632,  4.73684211,
        5.26315789,  5.78947368,  6.31578947,  6.84210526,  7.36842105,
        7.89473684,  8.42105263,  8.94736842,  9.47368421, 10.        ])
```

In [50]:
```
y
```

Out[50]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1.])
```

In [48]:
```
plt.scatter(x, y, c=y)
```

Out[48]:
```
<matplotlib.collections.PathCollection at 0x1d9a45c7d90>
```



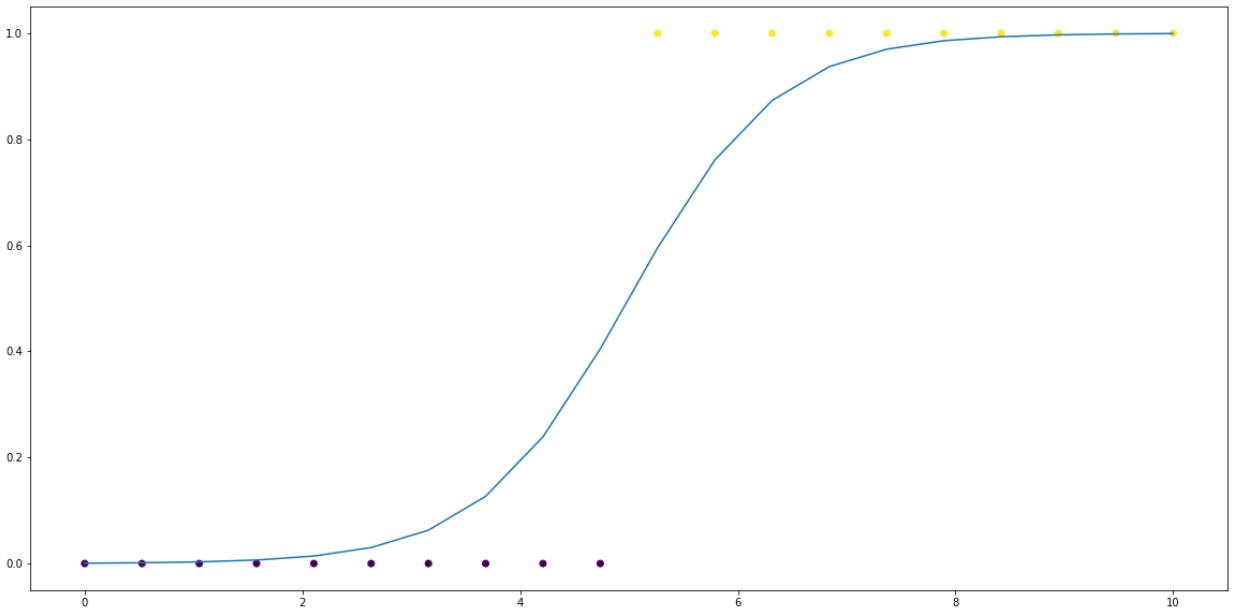
In [51]:
```
model = LogisticRegression()
```

In [52]:
```
model.fit(x.reshape(-1, 1),y)
```

Out[52]:
```
LogisticRegression()
```

In [55]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1))[:,1])
```

Out[55]:
```
[<matplotlib.lines.Line2D at 0x1d9a68399f0>]
```

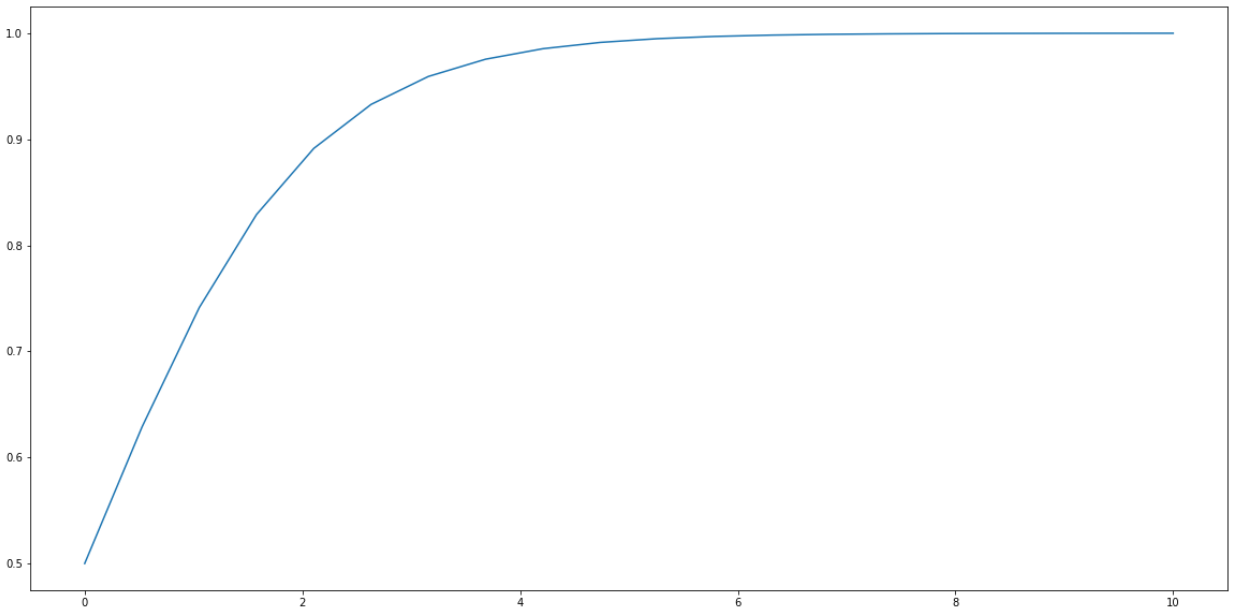


In [56]:
```
b, b0 = model.coef_, model.intercept_
model.coef_, model.intercept_
```

Out[56]:
```
(array([[1.46709085]]), array([-7.33542562]))
```

In [60]:
```
plt.plot(x, 1 / (1 + np.exp(-x)) )
```

Out[60]:
```
[<matplotlib.lines.Line2D at 0x1d9a6b44d90>]
```

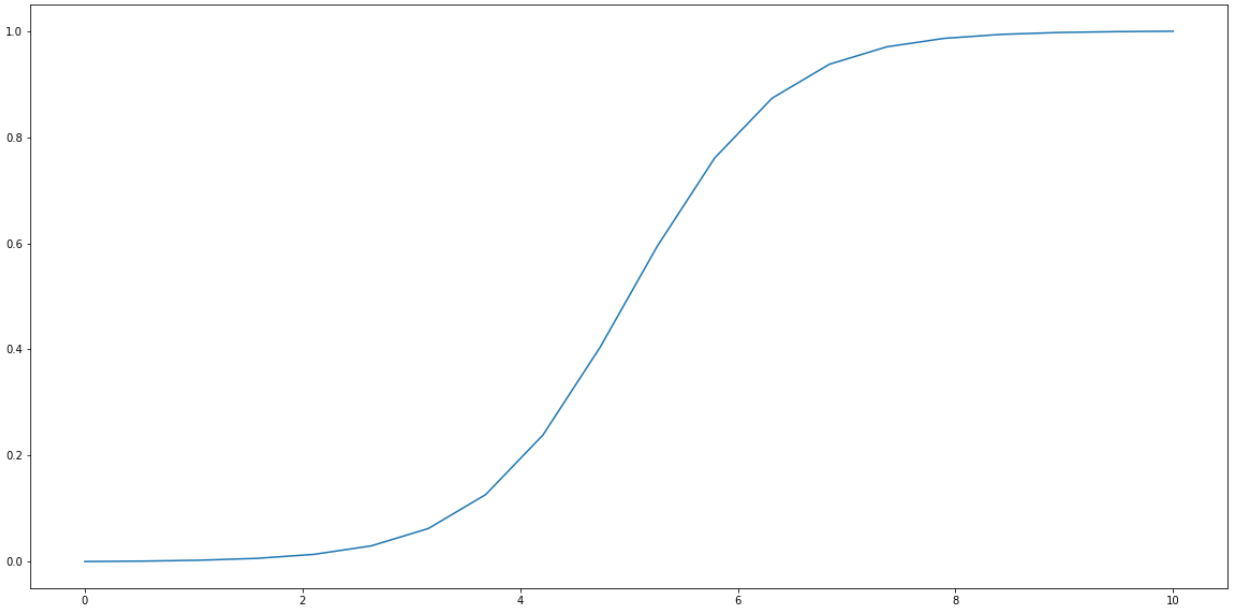


In [61]:
```
b
```

Out[61]:
```
array([[1.46709085]])
```

In [63]:
```
plt.plot(x, 1/(1+np.exp(-(b[0]*x +b0))))
```

Out[63]:
```
[<matplotlib.lines.Line2D at 0x1d9a6bdaa40>]
```



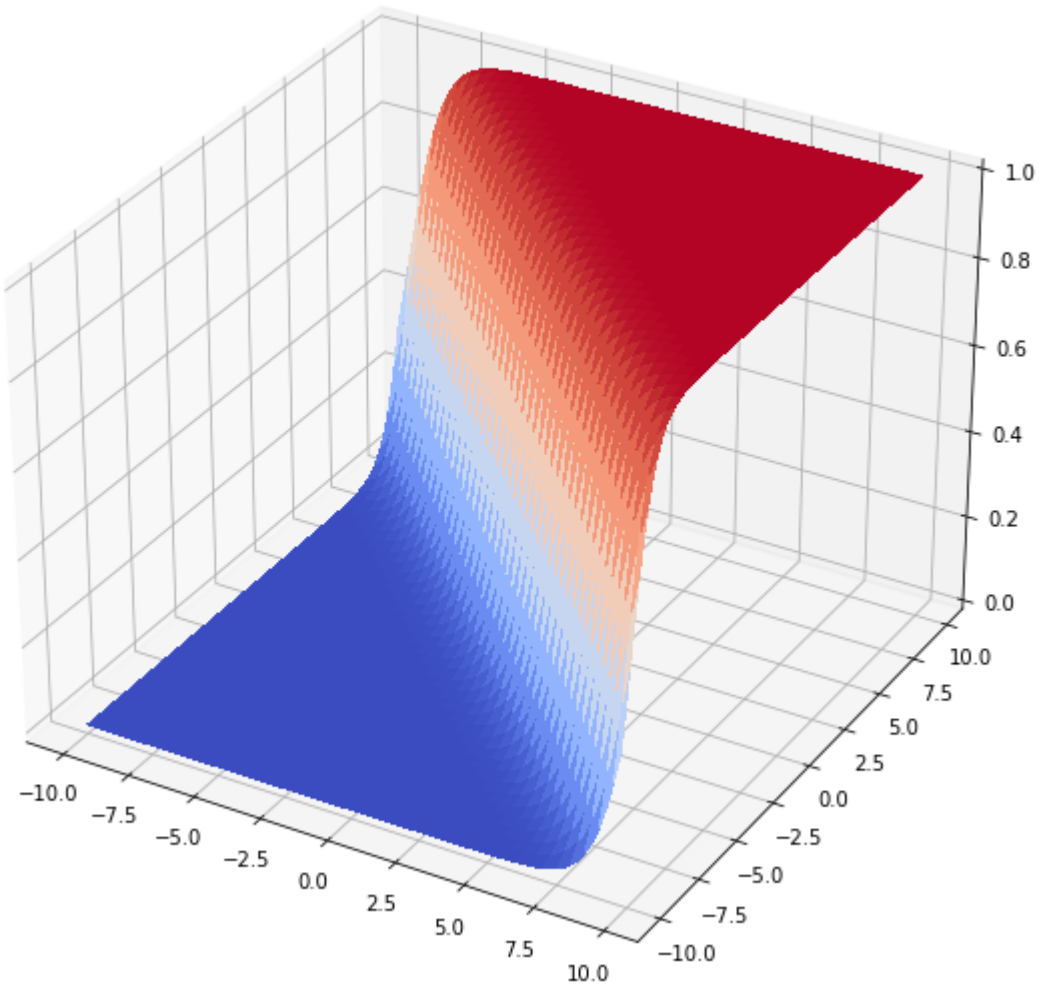
In [72]:
```
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import

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+7)))
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
```



In [73]:

```
X
```

Out[73]:
```
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 [74]:
```
Y
```

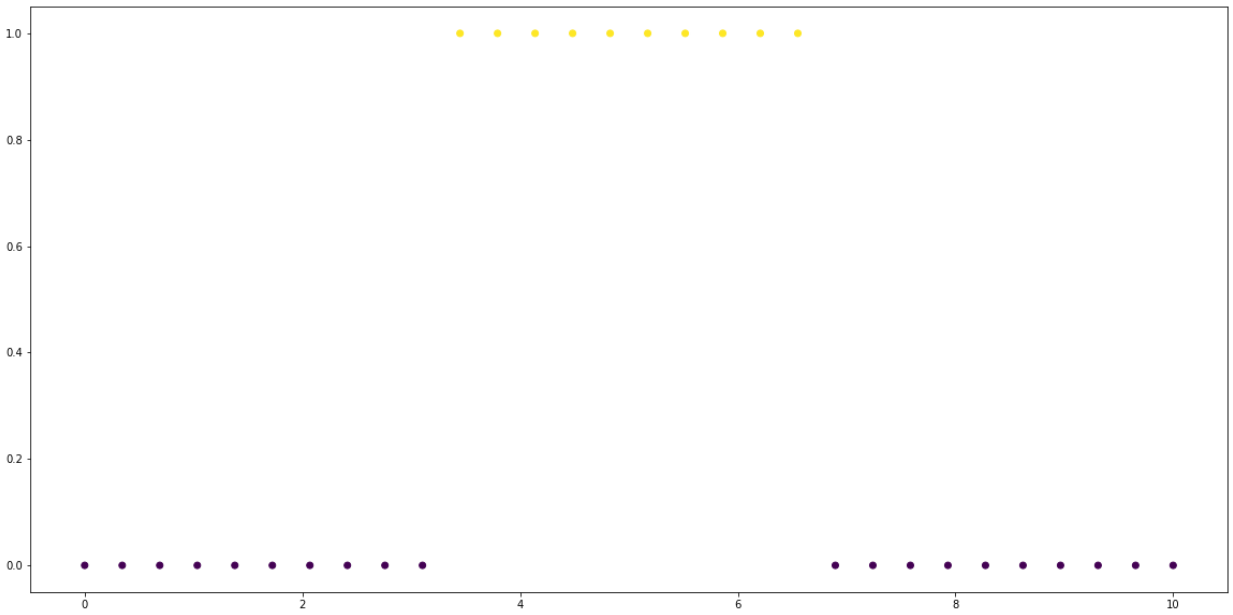
Out[74]:
```
array([[-10.  , -10.  , -10.  , ..., -10.  , -10.  , -10.  ],
       [ -9.75,  -9.75,  -9.75, ...,  -9.75,  -9.75,  -9.75],
       [ -9.5 ,  -9.5 ,  -9.5 , ...,  -9.5 ,  -9.5 ,  -9.5 ],
       ...,
       [  9.25,   9.25,   9.25, ...,   9.25,   9.25,   9.25],
       [  9.5 ,   9.5 ,   9.5 , ...,   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 [75]:
```
y = np.concatenate([np.zeros(10), np.ones(10), np.zeros(10)])
x = np.linspace(0, 10, len(y))
```

In [76]:
```
plt.scatter(x,y, c=y)
```

Out[76]:
```
<matplotlib.collections.PathCollection at 0x1d9aa7ad2a0>
```

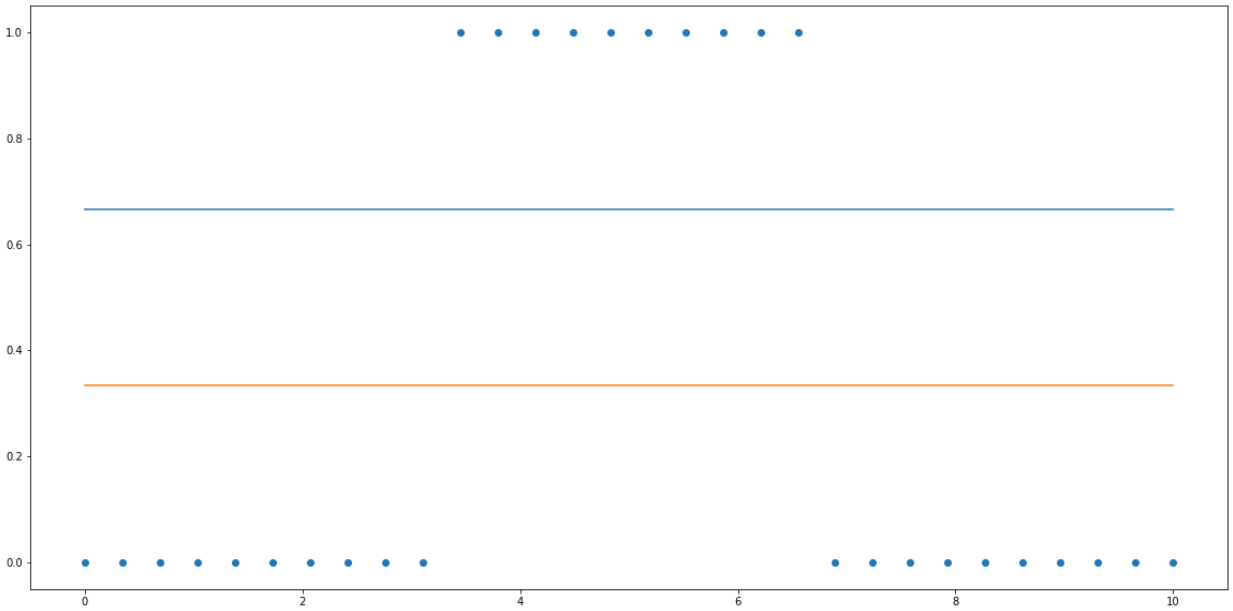


In [77]:
```
model.fit(x.reshape(-1, 1),y)
```

Out[77]:
```
LogisticRegression()
```

In [78]:
```
plt.scatter(x,y)
plt.plot(x, model.predict_proba(x.reshape(-1, 1)))
```

Out[78]:
```
[<matplotlib.lines.Line2D at 0x1d9a9ba8580>,
 <matplotlib.lines.Line2D at 0x1d9a9ba85e0>]
```



In [79]:
```
model1 = LogisticRegression()
model1.fit(x[:15].reshape(-1, 1),y[:15])
```

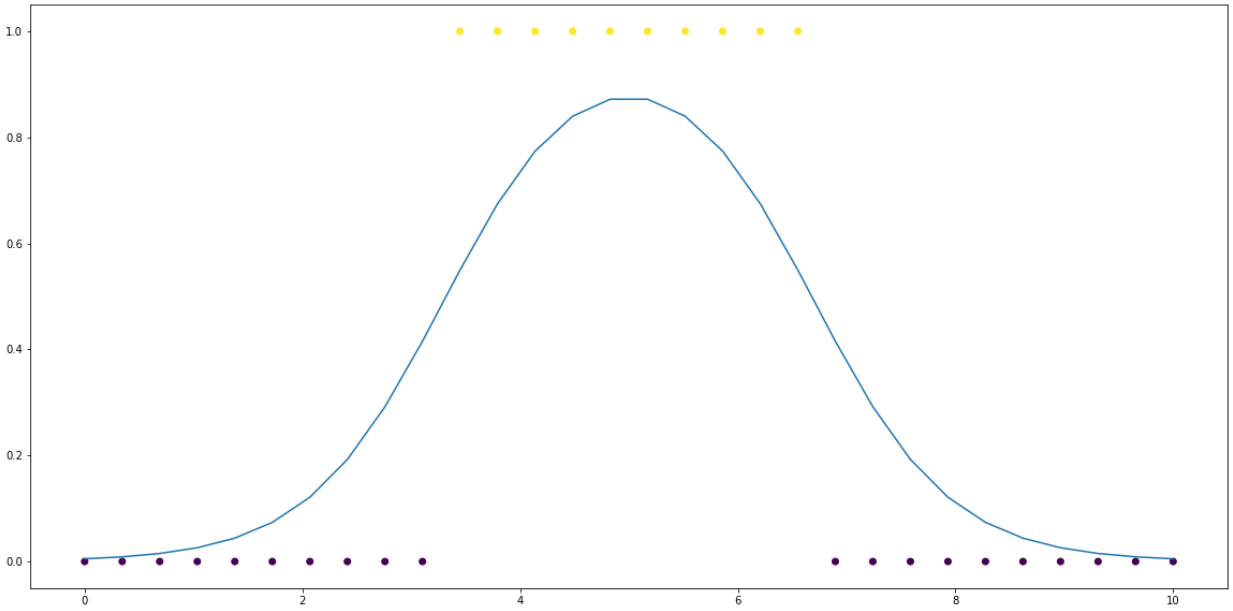
Out[79]:
```
LogisticRegression()
```

In [80]:
```
model2 = LogisticRegression()
model2.fit(x[15:].reshape(-1, 1),y[15:])
```

Out[80]:
```
LogisticRegression()
```

In [82]:
```
plt.scatter(x,y, c=y)
plt.plot(x, model1.predict_proba(x.reshape(-1, 1))[:,1] * model2.predict_proba(x.resh
```

Out[82]:
```
[<matplotlib.lines.Line2D at 0x1d9a9c42f20>]
```



In [83]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [84]:
```
from sklearn import preprocessing

enc = preprocessing.OrdinalEncoder()
```

In [85]:
```
transform_columns = ['sex', 'workclass', 'education', 'marital-status',
                     'occupation', 'relationship', 'race', 'sex',
                     'native-country', 'salary']
```

In [86]:
```
x = df.copy()

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

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

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

In [87]:
```
df.salary.unique()
```

Out[87]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [88]:
```
golden.salary.replace(' <=50K.', ' <=50K').replace(' >50K.', ' >50K').unique()
```

Out[88]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [89]:
```
model.fit(preprocessing.scale(x.drop('salary', axis=1)), x.salary)
```

Out[89]:
```
LogisticRegression()
```

In [90]:
```
pred = model.predict(preprocessing.scale(x.drop('salary', axis=1)))
pred_test = model.predict(preprocessing.scale(xt.drop('salary', axis=1)))
```

In [91]:
```
x.head()
```

Out[91]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	cap
0	39	7.0	77516	9.0	13	4.0	1.0	1.0	4.0	1.0	2
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 [92]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [93]:
```
accuracy_score(x.salary, pred)
```

Out[93]:
```
0.8250360861152913
```

In [94]:
```
confusion_matrix(x.salary, pred)
```

Out[94]:
```
array([[23300,  1420],
       [ 4277,  3564]], dtype=int64)
```

In [95]:
```
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 [96]:
```
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
```

Week 7 Assignment

In [183…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (accuracy_score,
                             classification_report,
                             confusion_matrix, auc, roc_curve
                            )
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
```

1. Use your own dataset (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`. Which algorithm is superior?

For my dataset, I will re-use the WineQT dataset that I found on Kaggle. Each row of the dataset is a type of wine and the features include details about the wine. Finally, the last feature in the dataset is quality -- I will use the other features to classify and predict the quality (1-10).

In [217…
```
wine = pd.read_csv('../data/WineQT.csv')
wine.head()
```

Out[217]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qu
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	

In [218…
```
x_train, x_test, y_train, y_test = train_test_split(wine.drop(['quality'], axis=1),
                                                    wine.quality, test_size=.25)
```

In [219…
```
model_log = LogisticRegression()
model_log.fit(preprocessing.scale(x_train), y_train)
```

Out[219]:
```
LogisticRegression()
```

In [221…
```
pred_log_test = model_log.predict(preprocessing.scale(x_test))
```

In [222…
```
print("Classification Report, Logistic Regression:")
print(classification_report(y_test, pred_log_test, labels = np.unique(pred_log_test)))
print("")
print("Confusion Matrix, Logistic Regression:")
confusion_matrix(y_test, pred_log_test, labels = np.unique(pred_log_test))
```

```
Classification Report, Logistic Regression:
              precision    recall  f1-score   support

           5       0.68      0.70      0.69       128
           6       0.50      0.64      0.57       107
           7       0.47      0.21      0.29        38
           8       0.00      0.00      0.00         2

   micro avg       0.58      0.60      0.59       275
   macro avg       0.41      0.39      0.39       275
weighted avg       0.58      0.60      0.58       275


Confusion Matrix, Logistic Regression:
```

Out[222]:
```
array([[89, 38,  1,  0],
       [29, 69,  8,  1],
       [ 4, 26,  8,  0],
       [ 0,  2,  0,  0]], dtype=int64)
```

In [223…
```
model_dt = DecisionTreeClassifier(criterion = "entropy", max_depth = 1)
model_dt.fit(x_train, y_train)
pred_dt_test = model_dt.predict(x_test)
```

In [224…
```
print("Classification Report, Decision Tree Classifier (max_depth = 1):")
print(classification_report(y_test, pred_dt_test, labels = np.unique(pred_dt_test)))
print("")
print("Confusion Matrix, Decision Tree Classifier (max_depth = 1):")
confusion_matrix(y_test, pred_dt_test, labels = np.unique(pred_dt_test))
```

```
Classification Report, Decision Tree Classifier (max_depth = 1):
              precision    recall  f1-score   support

           5       0.63      0.82      0.71       128
           6       0.49      0.54      0.51       107

   micro avg       0.57      0.69      0.63       235
   macro avg       0.56      0.68      0.61       235
weighted avg       0.56      0.69      0.62       235


Confusion Matrix, Decision Tree Classifier (max_depth = 1):
```

Out[224]:
```
array([[105,  23],
       [ 49,  58]], dtype=int64)
```

In [225…
```
model_dt2 = DecisionTreeClassifier(criterion = "entropy", max_depth = 3)
model_dt2.fit(x_train, y_train)
pred_dt2_test = model_dt2.predict(x_test)
```

In [226…

```
print("Classification Report, Decision Tree Classifier (max_depth = 3):")
print(classification_report(y_test, pred_dt2_test, labels = np.unique(pred_dt2_test)))
print("")
print("Confusion Matrix, Decision Tree Classifier (max_depth = 3):")
confusion_matrix(y_test, pred_dt2_test, labels = np.unique(pred_dt2_test))
```

```
Classification Report, Decision Tree Classifier (max_depth = 3):
              precision    recall  f1-score   support

           5       0.63      0.71      0.67       128
           6       0.46      0.52      0.49       107
           7       0.42      0.21      0.28        38

   micro avg       0.54      0.57      0.55       273
   macro avg       0.50      0.48      0.48       273
weighted avg       0.53      0.57      0.54       273


Confusion Matrix, Decision Tree Classifier (max_depth = 3):
```

Out[226]:

```
array([[91, 37,  0],
       [40, 56, 11],
       [ 4, 26,  8]], dtype=int64)
```

In [227…

```
print("Logistic Regression Accuracy Score:")
print(accuracy_score(y_test, pred_log_test))
print("")
print("Decision Tree Accuracy Score (max_depth = 1):")
print(accuracy_score(y_test, pred_dt_test))
print("")
print("Decision Tree Accuracy Score (max_depth = 3):")
print(accuracy_score(y_test, pred_dt2_test))
```

```
Logistic Regression Accuracy Score:
0.5804195804195804

Decision Tree Accuracy Score (max_depth = 1):
0.5699300699300699

Decision Tree Accuracy Score (max_depth = 3):
0.541958041958042
```

My Logistic Regression model outperforms the Decision Tree model with a max depth of 1, 2, and 3.

2. Repeat 1. but let the Decision Tree be much deeper to allow over-fitting. Compare the two models' test results again, and explain why it's superior

In [237…

```
print("Classification Report, Logistic Regression:")
print(classification_report(y_test, pred_log_test, labels = np.unique(pred_log_test)))
print("")
print("Confusion Matrix, Logistic Regression:")
confusion_matrix(y_test, pred_log_test, labels = np.unique(pred_log_test))
```

```
Classification Report, Logistic Regression:
              precision    recall  f1-score   support

           5       0.68      0.70      0.69       128
           6       0.50      0.64      0.57       107
           7       0.47      0.21      0.29        38
           8       0.00      0.00      0.00         2

   micro avg       0.58      0.60      0.59       275
   macro avg       0.41      0.39      0.39       275
weighted avg       0.58      0.60      0.58       275


Confusion Matrix, Logistic Regression:
```

Out[237]:
```
array([[89, 38,  1,  0],
       [29, 69,  8,  1],
       [ 4, 26,  8,  0],
       [ 0,  2,  0,  0]], dtype=int64)
```

In [241…
```
model_dt3 = DecisionTreeClassifier(criterion = "entropy", max_depth = 20)
model_dt3.fit(x_train, y_train)
pred_dt3_test = model_dt3.predict(x_test)
```

In [245…
```
print("Classification Report, Decision Tree Classifier (max_depth = 20):")
print(classification_report(y_test, pred_dt3_test, labels = np.unique(pred_dt3_test)))
print("")
print("Confusion Matrix, Decision Tree Classifier (max_depth = 20):")
confusion_matrix(y_test, pred_dt3_test, labels = np.unique(pred_dt3_test))
```

```
Classification Report, Decision Tree Classifier (max_depth = 20):
              precision    recall  f1-score   support

           4       0.00      0.00      0.00        10
           5       0.69      0.66      0.67       128
           6       0.54      0.60      0.57       107
           7       0.72      0.61      0.66        38
           8       0.50      1.00      0.67         2

   micro avg       0.61      0.61      0.61       285
   macro avg       0.49      0.57      0.51       285
weighted avg       0.61      0.61      0.61       285


Confusion Matrix, Decision Tree Classifier (max_depth = 20):
```

Out[245]:
```
array([[ 0,  5,  4,  1,  0],
       [ 5, 85, 38,  0,  0],
       [ 3, 31, 64,  8,  1],
       [ 0,  2, 12, 23,  1],
       [ 0,  0,  0,  0,  2]], dtype=int64)
```

In [246…
```
print("Logistic Regression Accuracy Score:")
print(accuracy_score(y_test, pred_log_test))
print("")
print("Decision Tree Accuracy Score (max_depth = 20):")
print(accuracy_score(y_test, pred_dt3_test))
```

```
Logistic Regression Accuracy Score:
0.5804195804195804

Decision Tree Accuracy Score (max_depth = 20):
0.6083916083916084
```

Once the decision tree model was overfit with a max depth of 20, the decision tree model slightly outperformed the logistic regression model. This makes sense as the overfit model introduces bias-variance tradeoff issues.

In [ ]: