Ben Roth Week 6 - Jupyter Notebook

March 22, 2020

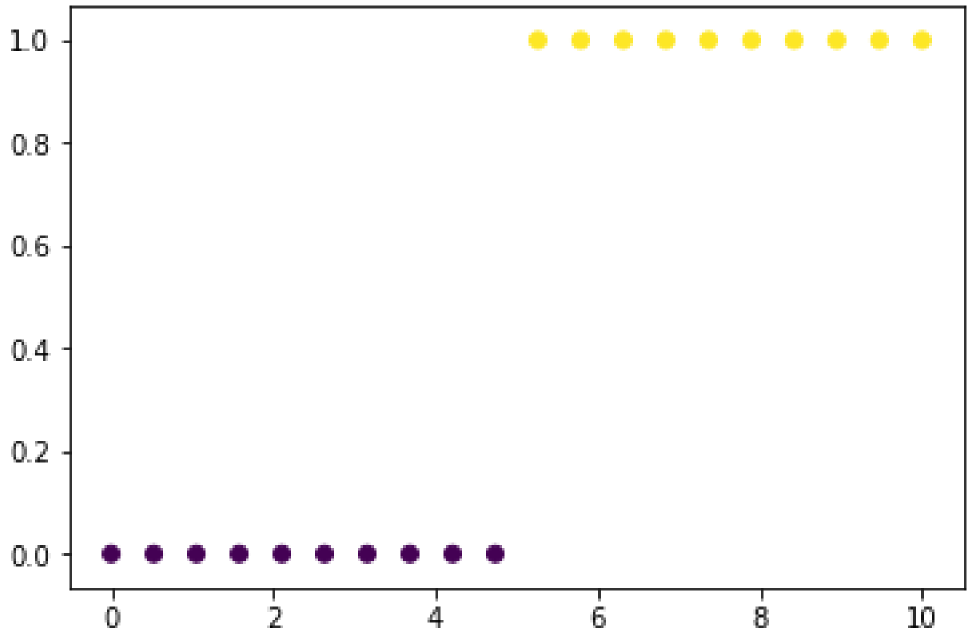
1 Assignment is at the bottom!

```
In [1]: 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
```

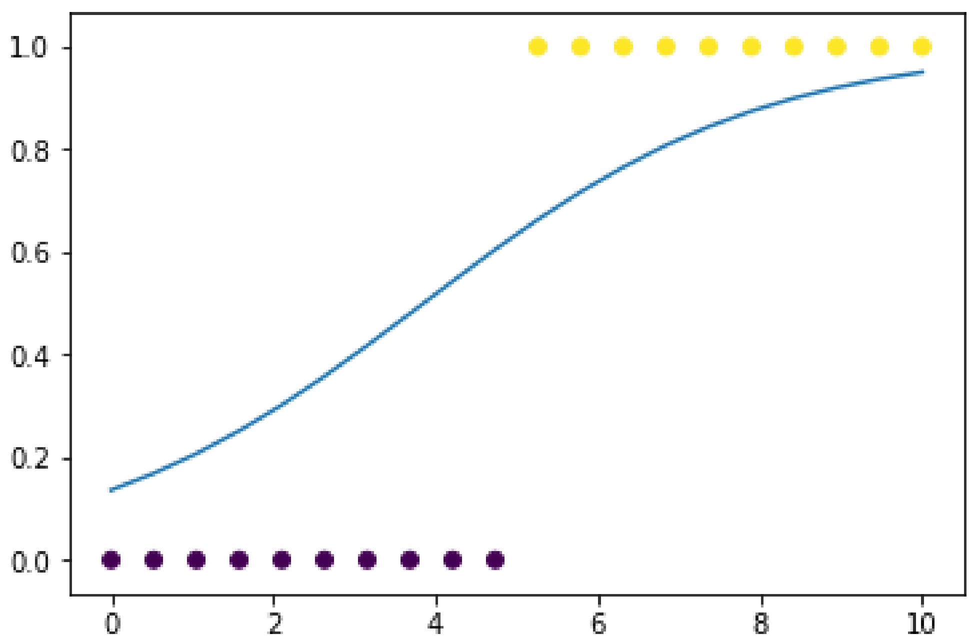
```
In [2]: y = np.concatenate([np.zeros(10), np.ones(10)])
        x = np.linspace(0, 10, len(y))
```

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

```
Out[3]: <matplotlib.collections.PathCollection at 0x1a1fff80f0>
```




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

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

/Users/Broth/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Futur
  FutureWarning)


Out[5]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)

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

Out[6]: [<matplotlib.lines.Line2D at 0x1a200e3358>]
```

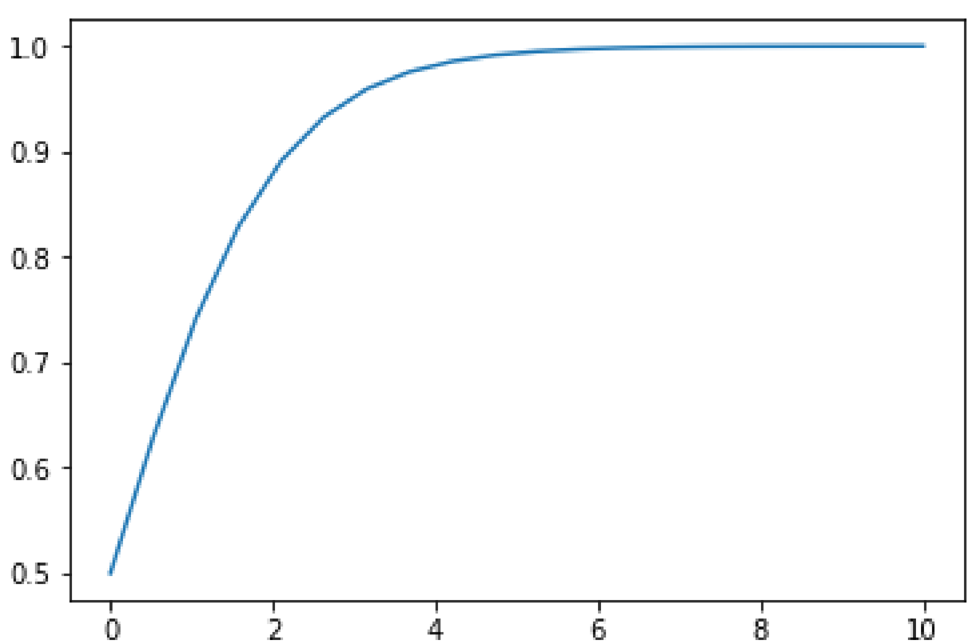


In [7]:
```
b, b0 = model.coef_, model.intercept_
model.coef_, model.intercept_
```

Out[7]: (array([[0.47990097]]), array([-1.84794266]))

In [8]:
```
plt.plot(x, 1/(1+np.exp(-x)))
```

Out[8]: [<matplotlib.lines.Line2D at 0x1a20588128>]

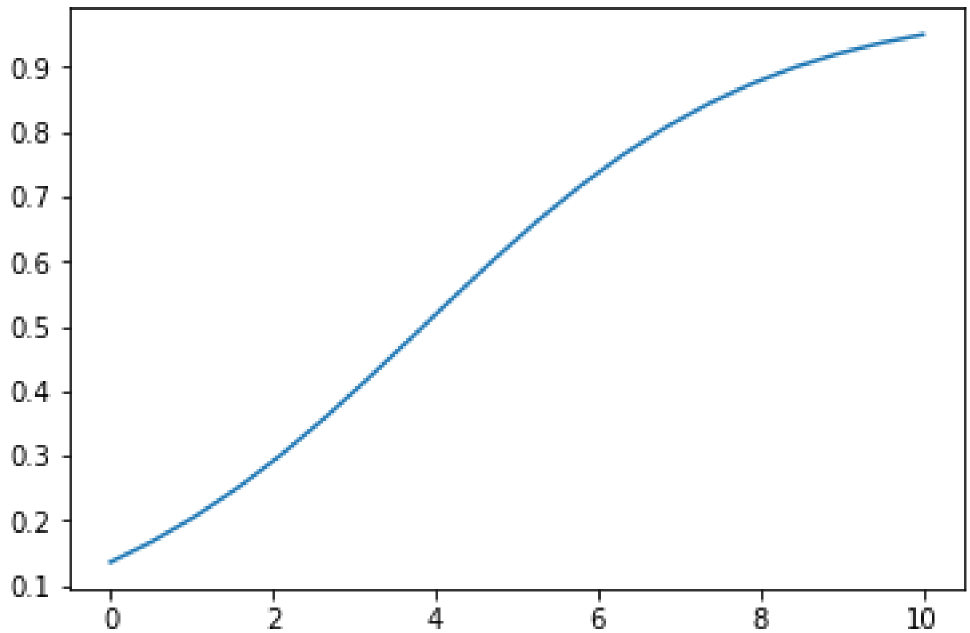


```
In [9]: b
```

```
Out[9]: array([[0.47990097]])
```

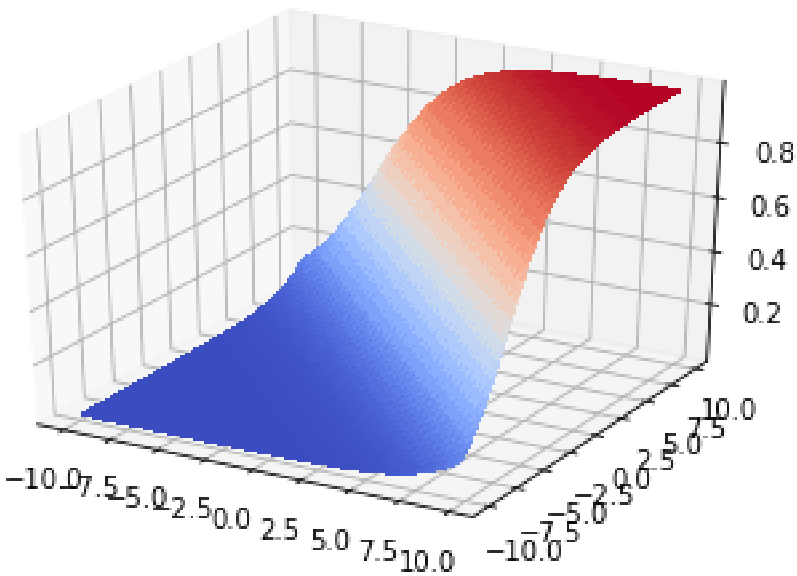
```
In [10]: plt.plot(x, 1/(1+np.exp(-(b[0]*x +b0))))
```

```
Out[10]: [<matplotlib.lines.Line2D at 0x1a2073a080>]
```




```
In [11]: 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)
```




```
In [12]: X
```

```
Out[12]: 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 [13]: Y
```

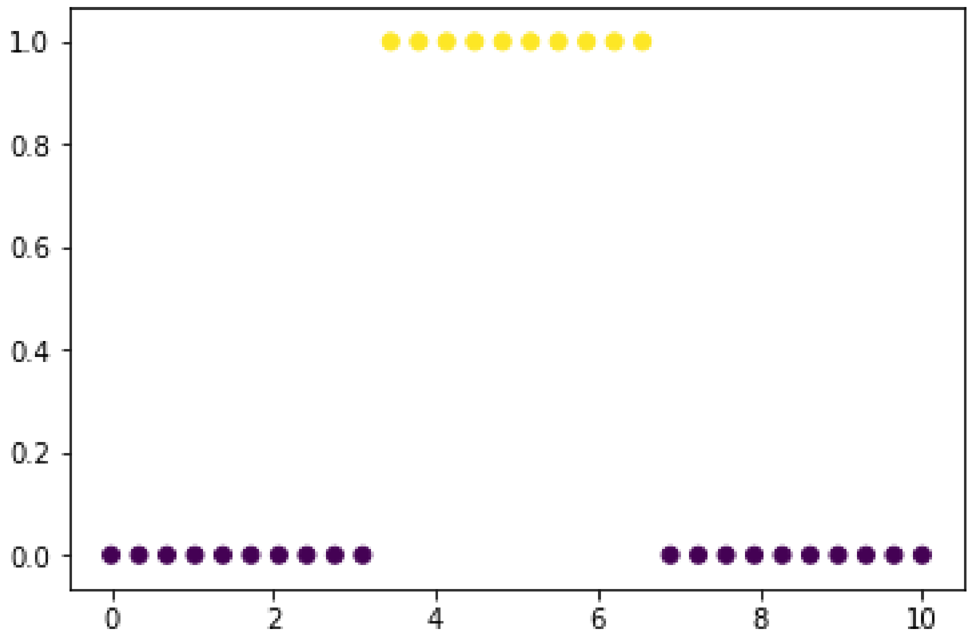
```
Out[13]: array([[-10.  , -10.  , -10.  , ..., -10.  , -10.  , -10.  ],
                [ -9.75,  -9.75,  -9.75, ...,  -9.75,  -9.75,  -9.75],
                [ -9.5 ,  -9.5 ,  -9.5 , ...,  -9.5 ,  -9.5 ,  -9.5 ],
                ...,
                [  9.25,   9.25,   9.25, ...,   9.25,   9.25,   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 0x1a20ae6f28>
```



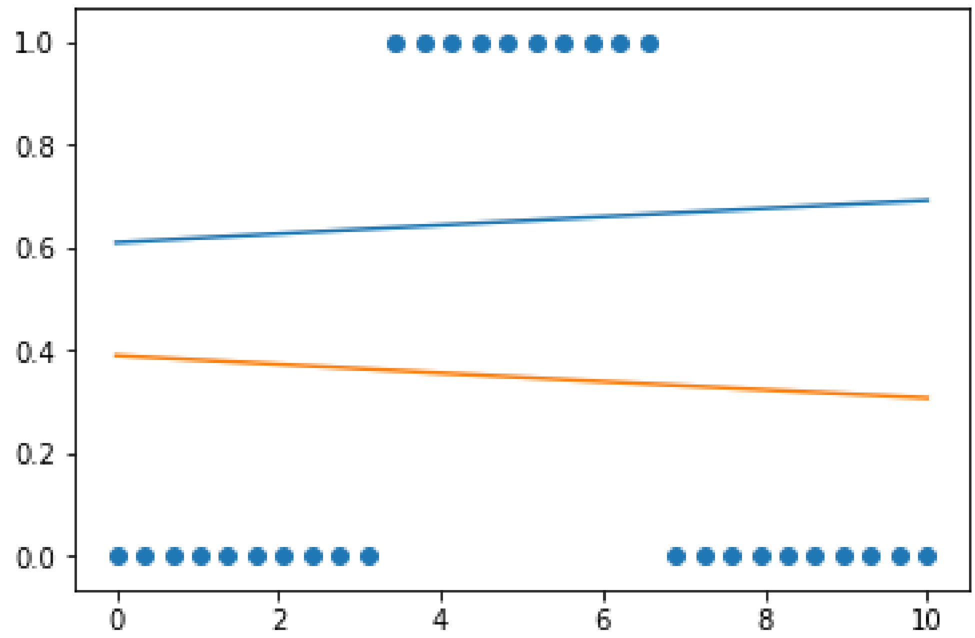

```
In [16]: model.fit(x.reshape(-1, 1),y)
```

```
/Users/Broth/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Future
  FutureWarning)
```

```
Out[16]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

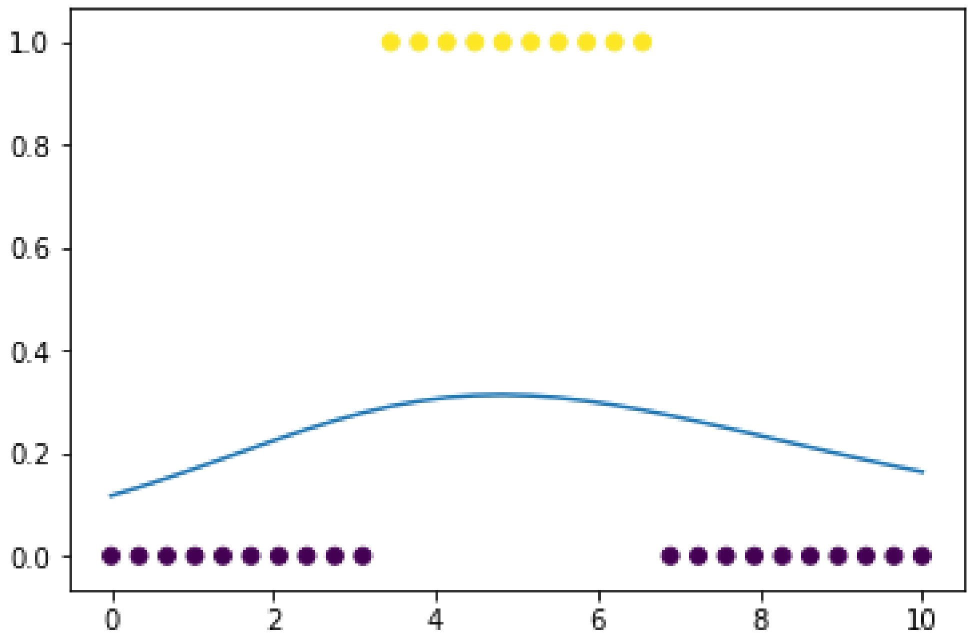
```
In [17]: plt.scatter(x,y)
         plt.plot(x, model.predict_proba(x.reshape(-1, 1)))
```

```
Out[17]: [<matplotlib.lines.Line2D at 0x1a20ae7668>,
          <matplotlib.lines.Line2D at 0x1a20a93860>]
```




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

/Users/Broth/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Future
  FutureWarning)


Out[18]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)

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

/Users/Broth/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Future
  FutureWarning)


Out[19]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)

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

```
Out[20]: [<matplotlib.lines.Line2D at 0x1a20894e48>]
```




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

In [ ]:

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(' >50K.', ' >50
         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').unique()
```

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

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

```
/Users/Broth/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: DataConversionWarni
  """Entry point for launching an IPython kernel.
/Users/Broth/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Future
  FutureWarning)
```

Out[27]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

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

```
/Users/Broth/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: DataConversionWarni
  """Entry point for launching an IPython kernel.
/Users/Broth/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: DataConversionWarni
```

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

Out[29]:
```
   age  workclass  fnlwgt  education  education-num  marital-status  \
0   39        7.0   77516        9.0             13             4.0
1   50        6.0   83311        9.0             13             2.0
2   38        4.0  215646       11.0              9             0.0
3   53        4.0  234721        1.0              7             2.0
4   28        4.0  338409        9.0             13             2.0

   occupation  relationship  race  sex  capital-gain  capital-loss  \
0         1.0           1.0   4.0  1.0          2174             0
1         4.0           0.0   4.0  1.0             0             0
2         6.0           1.0   4.0  1.0             0             0
3         6.0           0.0   2.0  1.0             0             0
4        10.0           5.0   2.0  0.0             0             0

   hours-per-week  native-country  salary
0              40            39.0     0.0
1              13            39.0     0.0
2              40            39.0     0.0
3              40            39.0     0.0
4              40             5.0     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]:
```
0.8250667977027732
```

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

Out[32]:
```
array([[23300,  1420],
       [ 4276,  3565]])
```

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

   micro avg       0.83      0.83      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

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

2 Assignment

2.1 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.

In [35]:
```
import random
random.seed(1234)
```

```
        df = pd.read_stata('../data/Titanic.dta')

        df.head(5)
```

```
Out[35]:    pclass  survived                                            name     sex  \
        0       1         1                    Allen, Miss. Elisabeth Walton  female
        1       1         1                   Allison, Master. Hudson Trevor    male
        2       1         0                     Allison, Miss. Helen Loraine  female
        3       1         0             Allison, Mr. Hudson Joshua Creighton    male
        4       1         0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  female

             age  sibsp  parch  ticket        fare    cabin embarked
        0  29.00      0      0   24160  211.337494       B5        S
        1   0.92      1      2  113781  151.550003  C22 C26        S
        2   2.00      1      2  113781  151.550003  C22 C26        S
        3  30.00      1      2  113781  151.550003  C22 C26        S
        4  25.00      1      2  113781  151.550003  C22 C26        S
```

```
In [36]: df = df.drop(['name', 'ticket', 'cabin'], axis = 1)

         from sklearn.preprocessing import LabelEncoder

         str_col = df.select_dtypes('object').columns

         for col in str_col:
             df[col] = LabelEncoder().fit_transform(df[col].values)
```

```
In [37]: df.isna().sum()
```

```
Out[37]: pclass        0
         survived      0
         sex           0
         age         263
         sibsp         0
         parch         0
         fare          1
         embarked      0
         dtype: int64
```

```
In [38]: df = df.dropna()
```

```
In [39]: X = df.drop('survived', axis = 1)
         y = df['survived']

         from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_stat
```

```
In [40]: from sklearn.tree import DecisionTreeClassifier
```

```
logit = LogisticRegression()

logit.fit(X_train, y_train)

tree = DecisionTreeClassifier(max_depth = 3)

tree.fit(X_train, y_train)
```

```
/Users/Broth/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Futur
  FutureWarning)
```

```
Out[40]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                        splitter='best')
```

```
In [41]: log_pred = logit.predict(X_test)
         tree1_pred = tree.predict(X_test)

         print('The classification report for the logistic regression model is:\n', classifica

         print('\nThe classification report for the shallow decision tree is:\n', classificati
```

```
The classification report for the logistic regression model is:
               precision    recall  f1-score   support

           0       0.88      0.82      0.85       222
           1       0.71      0.80      0.76       123

   micro avg       0.81      0.81      0.81       345
   macro avg       0.80      0.81      0.80       345
weighted avg       0.82      0.81      0.82       345


The classification report for the shallow decision tree is:
               precision    recall  f1-score   support

           0       0.89      0.81      0.85       226
           1       0.69      0.81      0.74       119

   micro avg       0.81      0.81      0.81       345
   macro avg       0.79      0.81      0.80       345
weighted avg       0.82      0.81      0.81       345
```

Looking at the two classification reports, the two models appear comprable, although the logit model performs slightly better.

2.2 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 [42]:
```
deep_tree = DecisionTreeClassifier(max_depth = 20)
deep_tree.fit(X_train, y_train)
deep_tree_preds = deep_tree.predict(X_test)
print('The classification report for the deep decision tree is:\n', classification_re

print('\nThe classification report for the logistic regression model is:\n', classifi
```

```
The classification report for the deep decision tree is:
               precision    recall  f1-score   support

           0       0.81      0.82      0.81       206
           1       0.72      0.71      0.72       139

   micro avg       0.77      0.77      0.77       345
   macro avg       0.77      0.76      0.76       345
weighted avg       0.77      0.77      0.77       345


The classification report for the logistic regression model is:
               precision    recall  f1-score   support

           0       0.88      0.82      0.85       222
           1       0.71      0.80      0.76       123

   micro avg       0.81      0.81      0.81       345
   macro avg       0.80      0.81      0.80       345
weighted avg       0.82      0.81      0.82       345
```

Comparing the deeper decision tree to the logistic regression model, the LR model clearly performs better. As suggested in the question, the deeper decision tree is over-fit