# Assignment is below at the end

- [https://scikit-learn.org/stable/modules/tree.html (https://scikit-learn.org/stable/modules/tree.html)](https://scikit-learn.org/stable/modules/tree.html)
- [https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)
- [https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html (https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html)](https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html)

In [1]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (20, 6)
plt.rcParams['font.size'] = 14
import pandas as pd
```

In [2]:

In [3]:

In [4]:

Out[4]:

| | age | workclass | fnlwgt | education | education-num | marital-status | occupation | relationship | race | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 25 | Private | 226802 | 11th | 7 | Never-married | Machine-op-inspct | Own-child | Black | |
| **1** | 38 | Private | 89814 | HS-grad | 9 | Married-civ-spouse | Farming-fishing | Husband | White | |
| **2** | 28 | Local-gov | 336951 | Assoc-acdm | 12 | Married-civ-spouse | Protective-serv | Husband | White | |
| **3** | 44 | Private | 160323 | Some-college | 10 | Married-civ-spouse | Machine-op-inspct | Husband | Black | |
| **4** | 18 | ? | 103497 | Some-college | 10 | Never-married | ? | Own-child | White | Fe |

In [5]:

Out[5]:

| | age | workclass | fnlwgt | education | education-num | marital-status | occupation | relationship | race |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White |
| **1** | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White |
| **2** | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White |
| **3** | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black |
| **4** | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black |

In [6]:

Out[6]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-coun
try',
       'salary'],
      dtype='object')
```

In [7]:

In [8]:

```
# Columns we want to transform
transform_columns = ['sex']

#Columns we can't use because non-numerical
non_num_columns = ['workclass', 'education', 'marital-status',
                   'occupation', 'relationship', 'race', 'sex',
```

## First let's try using `pandas.get_dummies()` to transform columns

In [9]:
```
dummies = pd.get_dummies(df[transform_columns])
```

Out[9]:

| | sex_ Female | sex_ Male |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 0 | 1 |
| **2** | 0 | 1 |
| **3** | 0 | 1 |
| **4** | 1 | 0 |
| **...** | ... | ... |
| **32556** | 1 | 0 |
| **32557** | 0 | 1 |
| **32558** | 1 | 0 |
| **32559** | 0 | 1 |
| **32560** | 1 | 0 |

32561 rows × 2 columns

In [10]:

Out[10]: (32561, 2)

# sklearn has a similar process for OneHot Encoding features

In [11]:
```
onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exi
```

```
/Users/test/opt/anaconda3/lib/python3.9/site-packages/sklearn/preproc
essing/_encoders.py:828: FutureWarning: `sparse` was renamed to `spar
se_output` in version 1.2 and will be removed in 1.4. `sparse_output`
is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

Out[11]:
```
▾                         OneHotEncoder
OneHotEncoder(handle_unknown='infrequent_if_exist', sparse=False,
              sparse_output=False)
```

In [12]:

Out[12]: [array([' Female', ' Male'], dtype=object)]

In [13]:
```
sex = onehot.transform(df[transform_columns])
```

Out[13]:
```
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       ...,
       [1., 0.],
       [0., 1.],
       [1., 0.]])
```

In [14]:

Out[14]:
```
(32561, 2)
```

## In addition to OneHot encoding there is Ordinal Encoding

In [15]:
```
enc = preprocessing.OrdinalEncoder()
enc.fit(df[["salary"]])
salary = enc.transform(df[["salary"]])
```

Out[15]:
```
array([[0.],
       [0.],
       [0.],
       ...,
       [0.],
       [0.],
       [1.]])
```

In [16]:

Out[16]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [17]:

```
x = df.copy()

# transformed = pd.get_dummies(df[transform_columns])


onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exi

enc = preprocessing.OrdinalEncoder()

enc.fit(df[["salary"]])


transformed = onehot.transform(df[transform_columns])
new_cols = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols)


x = pd.concat(
    [
        x.drop(non_num_columns, axis=1),
        df_trans
    ],
    axis=1,)
```

```
/Users/test/opt/anaconda3/lib/python3.9/site-packages/sklearn/preproc
essing/_encoders.py:828: FutureWarning: `sparse` was renamed to `spar
se_output` in version 1.2 and will be removed in 1.4. `sparse_output`
is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [18]:

Out[18]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 39 | 77516 | 13 | 2174 | 0 | 40 | 0.0 | 0.0 | 1.0 |
| **1** | 50 | 83311 | 13 | 0 | 0 | 13 | 0.0 | 0.0 | 1.0 |
| **2** | 38 | 215646 | 9 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 |
| **3** | 53 | 234721 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 |
| **4** | 28 | 338409 | 13 | 0 | 0 | 40 | 0.0 | 1.0 | 0.0 |

In [19]:
```
xt = golden.copy()

transformed = onehot.transform(xt[transform_columns])
new_cols = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols)

xt = pd.concat(
    [
        xt.drop(non_num_columns, axis=1),
        df_trans
    ],
    axis=1,)
```

In [20]:

Out[20]:
```
0.0    12435
1.0     3846
Name: salary, dtype: int64
```

In [21]:

Out[21]:
```
[array([' <=50K.', ' >50K.'], dtype=object)]
```

In [22]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

**Choose the model of your preference: DecisionTree or RandomForest**

In [23]:

In [24]:

In [25]:

Out[25]:
```
▾            DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [26]:

Out[26]:
```
8323
```

In [27]:

Out[27]:
```
[('age', 0.3246301235526168),
 ('education-num', 0.16031246022637352),
 ('capital-gain', 0.22768459768097968),
 ('capital-loss', 0.07879939734918551),
 ('hours-per-week', 0.15314756577839925),
 (' Female', 0.05457574683014615),
 (' Male', 0.0008501085822990509)]
```

In [28]:

Out[28]:
```
[('age', 0.3246301235526168),
 ('education-num', 0.16031246022637352),
 ('capital-gain', 0.22768459768097968),
 ('capital-loss', 0.07879939734918551),
 ('hours-per-week', 0.15314756577839925),
 (' Female', 0.05457574683014615),
 (' Male', 0.0008501085822990509)]
```

In [29]:

Out[29]:

| | age | education-num | capital-gain | capital-loss | hours-per-week | Female | Male |
|---|---|---|---|---|---|---|---|
| **0** | 39 | 13 | 2174 | 0 | 40 | 0.0 | 1.0 |
| **1** | 50 | 13 | 0 | 0 | 13 | 0.0 | 1.0 |
| **2** | 38 | 9 | 0 | 0 | 40 | 0.0 | 1.0 |
| **3** | 53 | 7 | 0 | 0 | 40 | 0.0 | 1.0 |
| **4** | 28 | 13 | 0 | 0 | 40 | 1.0 | 0.0 |

In [30]:

Out[30]:
```
set()
```

In [31]:

Out[31]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 ' Female',
 ' Male']
```

In [32]:
```
predictions = model.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [33]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [83]:
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/tq/y257w3hd2p59ywppnr10vgq40000gq/T/ipykernel_12750/4068432727.py in <module>
----> 1 accuracy_score(x.salary, predictions)

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/_param_validation.py in wrapper(*args, **kwargs)
    190
    191             try:
--> 192                 return func(*args, **kwargs)
    193             except InvalidParameterError as e:
    194                 # When the function is just a wrapper around an estimator, we allow

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py in accuracy_score(y_true, y_pred, normalize, sample_weight)
    219
    220     # Compute accuracy for each possible representation
--> 221     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    222     check_consistent_length(y_true, y_pred, sample_weight)
    223     if y_type.startswith("multilabel"):

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py in _check_targets(y_true, y_pred)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py in check_consistent_length(*arrays)
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:
--> 397         raise ValueError(
    398             "Found input variables with inconsistent numbers of samples: %r"
    399             % [int(l) for l in lengths]

ValueError: Found input variables with inconsistent numbers of samples: [32561, 16281]
```

In [36]:

Out[36]: 0.8210797862539156

In [37]:

Out[37]:
```
array([[11459,   976],
       [ 1937,  1909]])
```

In [38]:

```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89     12435
         1.0       0.66      0.50      0.57      3846

    accuracy                           0.82     16281
   macro avg       0.76      0.71      0.73     16281
weighted avg       0.81      0.82      0.81     16281
```

In [39]:

```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89     12435
         1.0       0.66      0.50      0.57      3846

    accuracy                           0.82     16281
   macro avg       0.76      0.71      0.73     16281
weighted avg       0.81      0.82      0.81     16281
```

In [40]:

Out[40]: 0.8955806025613464

In [41]:

Out[41]:
```
array([[24097,   623],
       [ 2777,  5064]])
```

In [42]:

```
              precision    recall  f1-score   support

         0.0       0.90      0.97      0.93     24720
         1.0       0.89      0.65      0.75      7841

    accuracy                           0.90     32561
   macro avg       0.89      0.81      0.84     32561
weighted avg       0.90      0.90      0.89     32561
```

In [43]:

```
              precision    recall  f1-score   support

         0.0       0.90      0.97      0.93     24720
         1.0       0.89      0.65      0.75      7841

    accuracy                           0.90     32561
   macro avg       0.89      0.81      0.84     32561
weighted avg       0.90      0.90      0.89     32561
```

## For the following use the above `adult` dataset.

## 1. Show the RandomForest outperforms the DecisionTree for a fixed `max_depth` by training using the train set and calculate `precision`, `recall`, `f1`, `confusion matrix` on golden-test set. Start with only numerical features/columns. (age, education-num, capital-gain, capital-loss, hours-per-week)

In [52]:

```python
#Data
x1 = x.copy()
```

In [53]:

```python
#Define numerical features
```

In [54]:

```python
#Create Decision Tree Models 1 and 2
x1_dt1 = DecisionTreeClassifier(criterion='entropy', max_depth = 7)
```

In [55]:

```python
#Create Random Forest Models 1 and 2
x1_rf1 = RandomForestClassifier(criterion='entropy', max_depth = 7)
```

In [56]:

```python
#Fit Random Forest Model 1 on Training Data x1 to predict Salary
```

Out[56]:

```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=7)
```

In [57]:
```
#Fit Decision Tree Model 1 on Training Data x1 to predict Salary
```

Out[57]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

In [58]:
```
#Fit Random Forest Model 2 on Training Data x1 to predict Salary
```

Out[58]:
```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=3)
```

In [59]:
```
#Fit Decision Tree Model 2 on Training Data x1 to predict Salary
```

Out[59]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [60]:
```
#Create a list of tuples with column names from x1 (minus Salary) and
#feature importance values for Random Forest Model 1
```

Out[60]:
```
[('age', 0.2450813417936444),
 ('fnlwgt', 0.010733328713049113),
 ('education-num', 0.21219465479619376),
 ('capital-gain', 0.2741767103722392),
 ('capital-loss', 0.06593908787163455),
 ('hours-per-week', 0.08475992435696231),
 (' Female', 0.05905772276560064),
 (' Male', 0.04805722933067612)]
```

In [61]:
```
#Create a list of tuples with column names from x1 (minus Salary) and
#feature importance values for Decision Tree Model 1
```

Out[61]:
```
[('age', 0.2829603491199833),
 ('fnlwgt', 0.006085062492757386),
 ('education-num', 0.19704210624159596),
 ('capital-gain', 0.3217591678808323),
 ('capital-loss', 0.05628692032464681 4),
 ('hours-per-week', 0.042718585499722656),
 (' Female', 0.09256167804785609),
 (' Male', 0.0005861303926053879)]
```

In [62]:
```
#Create a list of tuples with column names from x1 (minus Salary) and
#feature importance values for Random Forest Model 2
```

Out[62]:
```
[('age', 0.21721353930132772),
 ('fnlwgt', 0.0007535794857928688),
 ('education-num', 0.2140533418088539),
 ('capital-gain', 0.3002026247152713),
 ('capital-loss', 0.03988394636620356),
 ('hours-per-week', 0.08914690529914691),
 (' Female', 0.08015552318557428),
 (' Male', 0.05859053983782941)]
```

In [63]:
```
#Create a list of tuples with column names from x1 (minus Salary) and
#feature importance values for Decision Tree Model 2
```

Out[63]:
```
[('age', 0.34233309386332306),
 ('fnlwgt', 0.0),
 ('education-num', 0.22140116182053377),
 ('capital-gain', 0.4362657443161431),
 ('capital-loss', 0.0),
 ('hours-per-week', 0.0),
 (' Female', 0.0),
 (' Male', 0.0)]
```

In [82]:
```
#Create array of predicted Salary values for Random Forest Model 1
```

In [65]:
```
#Create array of predicted Salary values for Decision Tree Model 1
```

In [66]:
```
#Create array of predicted Salary values for Random Forest Model 2
```

In [67]:
```
#Create array of predicted Salary values for Decision Tree Model 2
```

In [68]:
```
#Calculate accuracy score between the predicted and actual salary valu
```

Out[68]:
```
0.8390762238191757
```

In [69]:
```
#Calculate accuracy score between the predicted and actual salary valu
```

Out[69]:
```
0.8309686137215159
```

In [70]:
```
#Calculate accuracy score between the predicted and actual salary valu
```

Out[70]:
```
0.8097782691480867
```

In [71]:
```
#Calculate accuracy score between the predicted and actual salary valu
```

Out[71]:
```
0.8031447699772741
```

In [72]:
```
#Create Confusion Matrix for Random Forest Model 1
```

Out[72]:
```
array([[11976,   459],
       [ 2161,  1685]])
```

In [73]:
```
#Create Confusion Matrix for Decision Tree Model 1
```

Out[73]:
```
array([[11767,   668],
       [ 2084,  1762]])
```

In [74]:
```
#Create Confusion Matrix for Random Forest Model 2
```

Out[74]:
```
array([[12417,    18],
       [ 3079,   767]])
```

In [75]:
```
#Create Confusion Matrix for Decision Tree Model 2
```

Out[75]:
```
array([[12428,     7],
       [ 3198,   648]])
```

In [76]:
```
#Generate Classification Rport with precision, recall, and f1-score fo
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.96      0.90     12435
         1.0       0.79      0.44      0.56      3846

    accuracy                           0.84     16281
   macro avg       0.82      0.70      0.73     16281
weighted avg       0.83      0.84      0.82     16281
```

In [77]:
```
#Generate Classification Rport with precision, recall, and f1-score fo
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.95      0.90     12435
         1.0       0.73      0.46      0.56      3846

    accuracy                           0.83     16281
   macro avg       0.79      0.70      0.73     16281
weighted avg       0.82      0.83      0.82     16281
```

In [78]:
```
#Generate Classification Rport with precision, recall, and f1-score fo
```

```
              precision    recall  f1-score   support

         0.0       0.80      1.00      0.89     12435
         1.0       0.98      0.20      0.33      3846

    accuracy                           0.81     16281
   macro avg       0.89      0.60      0.61     16281
weighted avg       0.84      0.81      0.76     16281
```

In [79]:
```
#Generate Classification Rport with precision, recall, and f1-score fo
```

```
              precision    recall  f1-score   support

         0.0       0.80      1.00      0.89     12435
         1.0       0.99      0.17      0.29      3846

    accuracy                           0.80     16281
   macro avg       0.89      0.58      0.59     16281
weighted avg       0.84      0.80      0.74     16281
```

#As shown above, the Random Forest Models are superior to their corresponding Decision

Tree Models for Accuracy, Precision, Recall, and F1

# 2. Use a RandomForest or DecisionTree and the `adult` dataset, systematically add new columns, one by one, that are non-numerical but converted using the feature-extraction techniques we learned. Using the golden-test set show `[precision, recall, f1, confusion matrix]` for each additional feature added.

In [84]:
```
#Create x2 from x1, encode Marital Status as integers
x2 = x1.copy()
x2['marital-status'] = enc.fit_transform(df[['marital-status']])
```

Out[84]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 39 | 77516 | 13 | 2174 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 |
| 1 | 50 | 83311 | 13 | 0 | 0 | 13 | 0.0 | 0.0 | 1.0 | 2.0 |
| 2 | 38 | 215646 | 9 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 0.0 |
| 3 | 53 | 234721 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 2.0 |
| 4 | 28 | 338409 | 13 | 0 | 0 | 40 | 0.0 | 1.0 | 0.0 | 2.0 |

In [85]:
```
#Create xt2 from xt1, encode Marital Status as integers
xt2 = xt1.copy()
xt2['marital-status'] = enc.fit_transform(golden[['marital-status']])
```

Out[85]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 25 | 226802 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 |
| 1 | 38 | 89814 | 9 | 0 | 0 | 50 | 0.0 | 0.0 | 1.0 | 2.0 |
| 2 | 28 | 336951 | 12 | 0 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 |
| 3 | 44 | 160323 | 10 | 7688 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 |
| 4 | 18 | 103497 | 10 | 0 | 0 | 30 | 0.0 | 1.0 | 0.0 | 4.0 |

In [87]:
```
#Create Random Forest Model 3 and Decision Tree Model 3
rf3 = RandomForestClassifier(criterion='entropy', max_depth = 3)
```

In [88]:
```
#Fit Random Forest Model 3 on x2 to predict Salary
```

In [89]:
```
#Create array of predicted Salary values for Random Forest Model 3
```

In [90]:
```
#Generate Classification Rport with precision, recall, and f1-score fo
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.98      0.90     12435
         1.0       0.87      0.34      0.49      3846

    accuracy                           0.83     16281
   macro avg       0.85      0.66      0.70     16281
weighted avg       0.84      0.83      0.80     16281
```

In [91]:
```
#Create x3 from x2, encode Native Country as integers
x3 = x2.copy()
x3['native-country'] = enc.fit_transform(df[['native-country']])
```

Out[91]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status | native-country |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 39 | 77516 | 13 | 2174 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 | 39.0 |
| **1** | 50 | 83311 | 13 | 0 | 0 | 13 | 0.0 | 0.0 | 1.0 | 2.0 | 39.0 |
| **2** | 38 | 215646 | 9 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 0.0 | 39.0 |
| **3** | 53 | 234721 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 2.0 | 39.0 |
| **4** | 28 | 338409 | 13 | 0 | 0 | 40 | 0.0 | 1.0 | 0.0 | 2.0 | 5.0 |

In [92]:
```
#Create xt3 from xt2, encode Native Country as integers
xt3 = xt2.copy()
xt3['native-country'] = enc.fit_transform(golden[['native-country']])
```

Out[92]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status | native-country |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 25 | 226802 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 | 38.0 |
| **1** | 38 | 89814 | 9 | 0 | 0 | 50 | 0.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **2** | 28 | 336951 | 12 | 0 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **3** | 44 | 160323 | 10 | 7688 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **4** | 18 | 103497 | 10 | 0 | 0 | 30 | 0.0 | 1.0 | 0.0 | 4.0 | 38.0 |

In [93]:
```
# Fit Random Forest Model 3 on x3 to Predict Salary, Store Predictions
x3_rf = rf3.fit(x3.drop(['salary'], axis=1), x3.salary)
x3_rf_pred = x3_rf.predict(xt3.drop(['salary'], axis=1))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.99      0.90     12435
         1.0       0.94      0.27      0.42      3846

    accuracy                           0.82     16281
   macro avg       0.88      0.63      0.66     16281
weighted avg       0.84      0.82      0.78     16281
```

In [94]:
```
#Create x4 from x3, encode Occupation as integers
x4 = x3.copy()
x4['occupation'] = enc.fit_transform(df[['occupation']])
```

Out[94]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status | native-country |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 39 | 77516 | 13 | 2174 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 | 39.0 |
| **1** | 50 | 83311 | 13 | 0 | 0 | 13 | 0.0 | 0.0 | 1.0 | 2.0 | 39.0 |
| **2** | 38 | 215646 | 9 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 0.0 | 39.0 |
| **3** | 53 | 234721 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 2.0 | 39.0 |
| **4** | 28 | 338409 | 13 | 0 | 0 | 40 | 0.0 | 1.0 | 0.0 | 2.0 | 5.0 |

In [95]:
```
#Create xt4 from xt3, encode Occupation as integers
xt4 = xt3.copy()
xt4['occupation'] = enc.fit_transform(golden[['occupation']])
```

Out[95]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status | native-country |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 25 | 226802 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 | 38.0 |
| **1** | 38 | 89814 | 9 | 0 | 0 | 50 | 0.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **2** | 28 | 336951 | 12 | 0 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **3** | 44 | 160323 | 10 | 7688 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **4** | 18 | 103497 | 10 | 0 | 0 | 30 | 0.0 | 1.0 | 0.0 | 4.0 | 38.0 |

In [96]:
```
# Fit Random Forest Model 3 on x4 to Predict Salary, Store Predictions
x4_rf = rf3.fit(x4.drop(['salary'], axis=1), x4.salary)
x4_rf_pred = x4_rf.predict(xt4.drop(['salary'], axis=1))
```

```
              precision    recall  f1-score   support

         0.0       0.82      0.99      0.90     12435
         1.0       0.89      0.29      0.44      3846

    accuracy                           0.82     16281
   macro avg       0.85      0.64      0.67     16281
weighted avg       0.84      0.82      0.79     16281
```

In [97]:
```
#Create x5 from x4, encode Workclass as integers
x5 = x4.copy()
x5['workclass'] = enc.fit_transform(df[['workclass']])
```

Out[97]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status | native-country |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 39 | 77516 | 13 | 2174 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 | 39.0 |
| **1** | 50 | 83311 | 13 | 0 | 0 | 13 | 0.0 | 0.0 | 1.0 | 2.0 | 39.0 |
| **2** | 38 | 215646 | 9 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 0.0 | 39.0 |
| **3** | 53 | 234721 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 2.0 | 39.0 |
| **4** | 28 | 338409 | 13 | 0 | 0 | 40 | 0.0 | 1.0 | 0.0 | 2.0 | 5.0 |

In [98]:
```
#Create xt5 from xt4, encode Workclass as integers
xt5 = xt4.copy()
xt5['workclass'] = enc.fit_transform(golden[['workclass']])
```

Out[98]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week | salary | Female | Male | marital-status | native-country |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 25 | 226802 | 7 | 0 | 0 | 40 | 0.0 | 0.0 | 1.0 | 4.0 | 38.0 |
| **1** | 38 | 89814 | 9 | 0 | 0 | 50 | 0.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **2** | 28 | 336951 | 12 | 0 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **3** | 44 | 160323 | 10 | 7688 | 0 | 40 | 1.0 | 0.0 | 1.0 | 2.0 | 38.0 |
| **4** | 18 | 103497 | 10 | 0 | 0 | 30 | 0.0 | 1.0 | 0.0 | 4.0 | 38.0 |

In [99]:
```
# Fit Random Forest Model 3 on x5 to Predict Salary, Store Predictions
x5_rf = rf3.fit(x5.drop(['salary'], axis=1), x5.salary)
x5_rf_pred = x5_rf.predict(xt5.drop(['salary'], axis=1))
```

```
              precision    recall  f1-score   support

         0.0       0.80      1.00      0.89     12435
         1.0       0.98      0.20      0.34      3846

    accuracy                           0.81     16281
   macro avg       0.89      0.60      0.61     16281
weighted avg       0.85      0.81      0.76     16281
```

#Converting columns to integers one by one actually reduced accuracy from 83 to 82 to 81 for Random Forest Model 3.

In [ ]: