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/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)

In [164]:
```
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 [165]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
```

In [166]:
```
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [167]:
```
golden.head()
```

Out[167]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K.
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K.
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K.
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K.
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K.

In [168]:
```
df.head()
```

Out[168]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [169]:
```
df.columns
```

Out[169]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'salary'],
      dtype='object')
```

In [170]:
```
from sklearn import preprocessing
```

In [171]:
```
# 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',
                   'native-country']
```

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

In [172]:
```
dummies = pd.get_dummies(df[transform_columns])
dummies
```

Out[172]:

	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 [173]:
```
dummies.shape
```

Out[173]:
```
(32561, 2)
```

sklearn has a similar process for OneHot Encoding features

In [301]:
```
onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse_output=False)
onehot.fit(df[transform_columns])
```

Out[301]:
```
▾ OneHotEncoder
OneHotEncoder(handle_unknown='infrequent_if_exist', sparse_output=False)
```

In [291]:
```
onehot.categories_
```

Out[291]:
```
[array([' Female', ' Male'], dtype=object)]
```

In [176]:
```
sex = onehot.transform(df[transform_columns])
sex
```

Out[176]:
```
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       ...,
       [1., 0.],
       [0., 1.],
       [1., 0.]])
```

In [177]:
```
sex.shape
```

Out[177]:
```
(32561, 2)
```

In addition to OneHot encoding there is Ordinal Encoding

In [178]:
```
enc = preprocessing.OrdinalEncoder()
enc.fit(df[["salary"]])
salary = enc.transform(df[["salary"]])
salary
```

Out[178]:
```
array([[0.],
       [0.],
       [0.],
       ...,
       [0.],
       [0.],
       [1.]])
```

In [179]:
```
enc.categories_[0]
```

Out[179]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [180]:
```
x = df.copy()

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


onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse_output=False).fit(df[transform_columns

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,)


x["salary"] = enc.transform(df[["salary"]])
```

In [181]:
```
x.head()
```

Out[181]:

	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 [182]:
```
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,)

xt["salary"] = enc.fit_transform(golden[["salary"]])
```

In [183]:
```
xt.salary.value_counts()
```

Out[183]:
```
0.0    12435
1.0     3846
Name: salary, dtype: int64
```

In [184]:
```
enc.categories_
```

Out[184]:
```
[array([' <=50K.', ' >50K.'], dtype=object)]
```

In [185]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

Choose the model of your preference: DecisionTree or RandomForest

In [186]:
```
model = RandomForestClassifier(criterion='entropy')
```

In [187]:
```
model = DecisionTreeClassifier(criterion='entropy', max_depth=None)
```

In [188]:
```
model.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)
```

Out[188]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [189]:
```
model.tree_.node_count
```

Out[189]:
```
8313
```

In [190]:
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```

Out[190]:
```
[('age', 0.3226394602924674),
 ('education-num', 0.1616023519291502),
 ('capital-gain', 0.22748700351709567),
 ('capital-loss', 0.079214470333594),
 ('hours-per-week', 0.1537256624686072),
 (' Female', 0.0013013658419538756),
 (' Male', 0.05402968561713164)]
```

In [191]:
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```

Out[191]:
```
[('age', 0.3226394602924674),
 ('education-num', 0.1616023519291502),
 ('capital-gain', 0.22748700351709567),
 ('capital-loss', 0.079214470333594),
 ('hours-per-week', 0.1537256624686072),
 (' Female', 0.0013013658419538756),
 (' Male', 0.05402968561713164)]
```

In [192]:
```
x.drop(['fnlwgt','salary'], axis=1).head()
```

Out[192]:

	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 [193]:
```
set(x.columns) - set(xt.columns)
```

Out[193]:
```
set()
```

In [194]:
```
list(x.drop('salary', axis=1).columns)
```

Out[194]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 ' Female',
 ' Male']
```

In [195]:
```
list(x)
```

Out[195]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'salary',
 ' Female',
 ' Male']
```

In [196]:
```
list(xt)
```

Out[196]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'salary',
 ' Female',
 ' Male']
```

In [197]:
```
predictions = model.predict(xt.drop(['fnlwgt','salary'], axis=1))
predictionsx = model.predict(x.drop(['fnlwgt','salary'], axis=1))
```

In [198]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [199]:
```
accuracy_score(xt.salary, predictions)
```

Out[199]:
```
0.8205269946563479
```

In [200]:
```
accuracy_score(xt.salary, predictions)
```

Out[200]:
```
0.8205269946563479
```

In [201]:
```
confusion_matrix(xt.salary, predictions)
```

Out[201]:
```
array([[11461,   974],
       [ 1948,  1898]])
```

In [202]:
```
print(classification_report(xt.salary, predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.89     12435
         1.0       0.66      0.49      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 [203]:
```
print(classification_report(xt.salary, predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.89     12435
         1.0       0.66      0.49      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 [204]:
```
accuracy_score(x.salary, predictionsx)
```

Out[204]:
```
0.8955806025613464
```

In [205]:
```
confusion_matrix(x.salary, predictionsx)
```

Out[205]:
```
array([[24097,   623],
       [ 2777,  5064]])
```

In [206]:
```
print(classification_report(x.salary, predictionsx))
```

```
              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 [207]:
```
print(classification_report(x.salary, predictionsx))
```

```
              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 [212]:
```
x1 = x.copy()
xt1 = xt.copy()
```

In [213]:
```
x1.head()
```

Out[213]:

	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 [214]:
```
xt1.head()
```

Out[214]:

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

In [271]:
```
x1_rf1 = RandomForestClassifier(criterion='entropy', max_depth = 7)
x1_dt1 = DecisionTreeClassifier(criterion='entropy', max_depth = 7)
x1_rf2 = RandomForestClassifier(criterion='entropy', max_depth = 3)
x1_dt2 = DecisionTreeClassifier(criterion='entropy', max_depth = 3)
```

In [258]:
```
x1_rf1.fit(x1.drop(['salary'],axis=1),x1.salary)
```

Out[258]:
```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=7)
```

In [259]:
```
x1_dt1.fit(x1.drop(['salary'],axis=1),x1.salary)
```

Out[259]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

In [272]:
```
x1_rf2.fit(x1.drop(['salary'],axis=1),x1.salary)
```

Out[272]:
```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=3)
```

In [273]:
```
x1_dt2.fit(x1.drop(['salary'],axis=1),x1.salary)
```

Out[273]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [260]:
```
list(zip(x1.drop(['salary'], axis=1).columns, x1_rf1.feature_importances_))
```

Out[260]:
```
[('age', 0.240093219743165),
 ('fnlwgt', 0.011534989957478658),
 ('education-num', 0.19846941504636445),
 ('capital-gain', 0.26679223958878184),
 ('capital-loss', 0.07051146866383945),
 ('hours-per-week', 0.09498045659518521),
 (' Female', 0.05217555194014917),
 (' Male', 0.06544265846503625)]
```

In [261]:
```
list(zip(x1.drop(['salary'], axis=1).columns, x1_dt1.feature_importances_))
```

Out[261]:
```
[('age', 0.2829603491199833),
 ('fnlwgt', 0.006306080004978193),
 ('education-num', 0.19704210624159596),
 ('capital-gain', 0.3217591678808323),
 ('capital-loss', 0.05606590281242601),
 ('hours-per-week', 0.042497567987501844),
 (' Female', 0.03501265095704486),
 (' Male', 0.05835617499563742)]
```

In [274]:
```
list(zip(x1.drop(['salary'], axis=1).columns, x1_rf2.feature_importances_))
```

Out[274]:
```
[('age', 0.24634078131354797),
 ('fnlwgt', 0.0007106486235433378),
 ('education-num', 0.1969002410895295),
 ('capital-gain', 0.28853155826174126),
 ('capital-loss', 0.04514355356598634),
 ('hours-per-week', 0.08644270766813066),
 (' Female', 0.05963320101352708),
 (' Male', 0.07629730846399385)]
```

In [275]:
```
list(zip(x1.drop(['salary'], axis=1).columns, x1_dt2.feature_importances_))
```

Out[275]:
```
[('age', 0.3423330938633231),
 ('fnlwgt', 0.0),
 ('education-num', 0.2214011618205338),
 ('capital-gain', 0.4357102902795111),
 ('capital-loss', 0.0),
 ('hours-per-week', 0.000555454036632138),
 (' Female', 0.0),
 (' Male', 0.0)]
```

In [262]:
```
rf1_pred1 = x1_rf1.predict(xt1.drop(['salary'],axis=1))
```

In [263]:
```
dt1_pred1 = x1_dt1.predict(xt1.drop(['salary'],axis=1))
```

In [276]:
```
rf2_pred2 = x1_rf2.predict(xt1.drop(['salary'],axis=1))
```

In [277]:
```
dt2_pred2 = x1_dt2.predict(xt1.drop(['salary'],axis=1))
```

In [264]:
```
accuracy_score(xt1.salary, rf1_pred1)
```

Out[264]:
```
0.8369879000061421
```

In [265]:
```
accuracy_score(xt1.salary, dt1_pred1)
```

Out[265]:
```
0.8309686137215159
```

In [ ]:
```
accuracy_score(xt1.salary, rf1_pred1)
```

In [ ]:
```
accuracy_score(xt1.salary, dt1_pred1)
```

In [266]:
```
confusion_matrix(xt1.salary, rf1_pred1)
```

Out[266]:
```
array([[12006,   429],
       [ 2225,  1621]])
```

In [267]:
```
confusion_matrix(xt1.salary, dt1_pred1)
```

Out[267]:
```
array([[11767,   668],
       [ 2084,  1762]])
```

In [278]:
```
confusion_matrix(xt1.salary, rf2_pred2)
```

Out[278]:
```
array([[12410,    25],
       [ 3032,   814]])
```

In [280]:
```
confusion_matrix(xt1.salary, dt2_pred2)
```

Out[280]:
```
array([[12428,     7],
       [ 3199,   647]])
```

In [279]:
```
print(classification_report(xt1.salary, rf1_pred1))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.97      0.90     12435
         1.0       0.79      0.42      0.55      3846

    accuracy                           0.84     16281
   macro avg       0.82      0.69      0.73     16281
weighted avg       0.83      0.84      0.82     16281
```

In [269]:
```
print(classification_report(xt1.salary, dt1_pred1))
```

```
              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 [281]:
```
print(classification_report(xt1.salary, rf2_pred2))
```

```
              precision    recall  f1-score   support

         0.0       0.80      1.00      0.89     12435
         1.0       0.97      0.21      0.35      3846

    accuracy                           0.81     16281
   macro avg       0.89      0.60      0.62     16281
weighted avg       0.84      0.81      0.76     16281
```

In [282]:
```
print(classification_report(xt1.salary, dt2_pred2))
```

```
              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
```

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 [296]:
```
x1.head()
```

Out[296]:

	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 [297]:
```
df.head()
```

Out[297]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [307]:
```
x2 = x1.copy()
x2['marital-status'] = enc.fit_transform(df[['marital-status']])
x2.head()
```

Out[307]:

	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 [309]:
```
xt2 = xt1.copy()
xt2['marital-status'] = enc.fit_transform(golden[['marital-status']])
xt2.head()
```

Out[309]:

	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 [311]:
```
rf3 = RandomForestClassifier(criterion='entropy', max_depth = 3)
dt3 = DecisionTreeClassifier(criterion='entropy', max_depth = 3)
```

In [314]:
```
x2_rf = rf3.fit(x2.drop(['salary'], axis=1), x2.salary)
```

In [321]:
```
x2_rf_pred = x2_rf.predict(xt2.drop(['salary'], axis=1))
```

In [322]:
```
print(classification_report(xt2.salary, x2_rf_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.98      0.90     12435
         1.0       0.86      0.33      0.48      3846

    accuracy                           0.83     16281
   macro avg       0.84      0.66      0.69     16281
weighted avg       0.83      0.83      0.80     16281
```

In [318]:
```
x3 = x2.copy()
x3['native-country'] = enc.fit_transform(df[['native-country']])
x3.head()
```

Out[318]:

	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 [325]:
```
xt3 = xt2.copy()
xt3['native-country'] = enc.fit_transform(golden[['native-country']])
xt3.head()
```

Out[325]:

	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 [326]:
```
x3_rf = rf3.fit(x3.drop(['salary'], axis=1), x3.salary)
x3_rf_pred = x3_rf.predict(xt3.drop(['salary'], axis=1))
print(classification_report(xt3.salary, x3_rf_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.81      1.00      0.89     12435
         1.0       0.98      0.22      0.37      3846

    accuracy                           0.82     16281
   macro avg       0.89      0.61      0.63     16281
weighted avg       0.85      0.82      0.77     16281
```

In [319]:
```
x4 = x3.copy()
x4['occupation'] = enc.fit_transform(df[['occupation']])
x4.head()
```

Out[319]:

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

In [327]:
```
xt4 = xt3.copy()
xt4['occupation'] = enc.fit_transform(golden[['occupation']])
xt4.head()
```

Out[327]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	Female	Male	marital-status	native-country	occupation
0	25	226802	7	0	0	40	0.0	0.0	1.0	4.0	38.0	7.0
1	38	89814	9	0	0	50	0.0	0.0	1.0	2.0	38.0	5.0
2	28	336951	12	0	0	40	1.0	0.0	1.0	2.0	38.0	11.0
3	44	160323	10	7688	0	40	1.0	0.0	1.0	2.0	38.0	7.0
4	18	103497	10	0	0	30	0.0	1.0	0.0	4.0	38.0	0.0

In [328]:
```
x4_rf = rf3.fit(x4.drop(['salary'], axis=1), x4.salary)
x4_rf_pred = x4_rf.predict(xt4.drop(['salary'], axis=1))
print(classification_report(xt4.salary, x4_rf_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.81      1.00      0.89     12435
         1.0       0.98      0.22      0.36      3846

    accuracy                           0.82     16281
   macro avg       0.89      0.61      0.63     16281
weighted avg       0.85      0.82      0.77     16281
```

In [320]:
```
x5 = x4.copy()
x5['workclass'] = enc.fit_transform(df[['workclass']])
x5.head()
```

Out[320]:

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

In [329]:
```
xt5 = xt4.copy()
xt5['workclass'] = enc.fit_transform(golden[['workclass']])
xt5.head()
```

Out[329]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	Female	Male	marital-status	native-country	occupation	workclass
0	25	226802	7	0	0	40	0.0	0.0	1.0	4.0	38.0	7.0	4.0
1	38	89814	9	0	0	50	0.0	0.0	1.0	2.0	38.0	5.0	4.0
2	28	336951	12	0	0	40	1.0	0.0	1.0	2.0	38.0	11.0	2.0
3	44	160323	10	7688	0	40	1.0	0.0	1.0	2.0	38.0	7.0	4.0
4	18	103497	10	0	0	30	0.0	1.0	0.0	4.0	38.0	0.0	0.0

In [330]:
```
x5_rf = rf3.fit(x5.drop(['salary'], axis=1), x5.salary)
x5_rf_pred = x5_rf.predict(xt5.drop(['salary'], axis=1))
print(classification_report(xt5.salary, x5_rf_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.80      1.00      0.89     12435
         1.0       0.98      0.21      0.34      3846

    accuracy                           0.81     16281
   macro avg       0.89      0.60      0.62     16281
weighted avg       0.85      0.81      0.76     16281
```