Assignment 6: Coder

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 [9]:
```
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
from sklearn import preprocessing
enc = preprocessing.OrdinalEncoder()
```

In [10]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
```

In [11]:
```
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [12]:
```
df.columns
```

Out[12]:
```
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 [13]:
```
from sklearn import preprocessing
```

In [14]:
```
# 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']
```

In [15]:
```
x = df.copy()

x = pd.concat([x.drop(non_num_columns, axis=1),
               pd.get_dummies(df[transform_columns])], axis=1)
```

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

In [16]:
```
x.head()
```

Out[16]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male
0	39	77516	13	2174	0	40	0.0	False	True
1	50	83311	13	0	0	13	0.0	False	True
2	38	215646	9	0	0	40	0.0	False	True
3	53	234721	7	0	0	40	0.0	False	True
4	28	338409	13	0	0	40	0.0	True	False

In [17]:
```
xt = golden.copy()

xt = pd.concat([xt.drop(non_num_columns, axis=1),
               pd.get_dummies(golden[transform_columns])], axis=1)

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

In [18]:
```
enc.categories_
```

Out[18]:
```
[array([' <=50K.', ' >50K.'], dtype=object)]
```

In [19]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

Fit Models to Random Forest Classifier ("model") and Decision Tree Classifier ("model2")

In [20]:
```
model = RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [21]:
```
model2 = DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [22]:
```
model.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)
```

Out[22]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [23]:
```
model2.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)
```

Out[23]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [24]:
```
model2.tree_.node_count
```
Out[24]:
```
49
```

Random Forest Importances

In [25]:
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```
Out[25]:
```
[('age', 0.22933568842746035),
 ('education-num', 0.215989134735358),
 ('capital-gain', 0.2975563756568905),
 ('capital-loss', 0.051404983439975266),
 ('hours-per-week', 0.0829058608579065),
 ('sex_ Female', 0.06535516049709131),
 ('sex_ Male', 0.05745279638531799)]
```

Decision Tree Importances

In [26]:
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```
Out[26]:
```
[('age', 0.22933568842746035),
 ('education-num', 0.215989134735358),
 ('capital-gain', 0.2975563756568905),
 ('capital-loss', 0.051404983439975266),
 ('hours-per-week', 0.0829058608579065),
 ('sex_ Female', 0.06535516049709131),
 ('sex_ Male', 0.05745279638531799)]
```
In [27]:
```
#RF Predictions
predictions = model.predict(xt.drop(['fnlwgt','salary'], axis=1))
predictionsx = model.predict(x.drop(['fnlwgt','salary'], axis=1))
```
In [28]:
```
#DT Predictions
predictions2 = model2.predict(xt.drop(['fnlwgt','salary'], axis=1))
predictionsx2 = model2.predict(x.drop(['fnlwgt','salary'], axis=1))
```
In [29]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

Random Forest vs Decision Tree Accuracy Scores

In [30]:
```
accuracy_score(xt.salary, predictions)
```
Out[30]:
```
0.83078434985566
```
In [31]:
```
accuracy_score(xt.salary, predictions2)
```

Out[31]: 0.8200970456360175

Random Forest vs Decision Tree Confusion Matrices

In [32]:
```
confusion_matrix(xt.salary, predictions)
```

Out[32]:
```
array([[12047,   388],
       [ 2367,  1479]], dtype=int64)
```

In [33]:
```
confusion_matrix(xt.salary, predictions2)
```

Out[33]:
```
array([[11458,   977],
       [ 1952,  1894]], dtype=int64)
```

Precision, Recall, F1-Score Comparison: Random Forest versus Decision Tree

In [34]:
```
print(classification_report(xt.salary, predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.97      0.90     12435
         1.0       0.79      0.38      0.52      3846

    accuracy                           0.83     16281
   macro avg       0.81      0.68      0.71     16281
weighted avg       0.83      0.83      0.81     16281
```

In [35]:
```
print(classification_report(xt.salary, predictions2))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.89     12435
         1.0       0.66      0.49      0.56      3846

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

Comparing the classification reports, we see that when max_depth is set to 5, compared to the DecisionTree model (bottom) the RandomForest model (top) has comparable precision and f1 score on "0.0", but has substantially higher recall on "0.0" and higher precision on "1.0". RandomForest actually has lower recall and f1-score on "1.0", but slightly higher overall accuracy.

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.

Add "Race"

In [36]:
```
# Columns we want to transform
transform_columns = ['sex', 'race']

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

In [37]:
```
x2 = df.copy()

x2 = pd.concat([x2.drop(non_num_columns, axis=1),
               pd.get_dummies(df[transform_columns])], axis=1)

x2["salary"] = enc.fit_transform(df[["salary"]])
```

In [38]:
```
xt2 = golden.copy()

xt2 = pd.concat([xt2.drop(non_num_columns, axis=1),
               pd.get_dummies(golden[transform_columns])], axis=1,)

xt2["salary"] = enc.fit_transform(golden[["salary"]])
```

In [39]:
```
x2.head()
```

Out[39]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	race_Amer-Indian-Eskimo	ra As I Islar
0	39	77516	13	2174	0	40	0.0	False	True	False	F
1	50	83311	13	0	0	13	0.0	False	True	False	F
2	38	215646	9	0	0	40	0.0	False	True	False	F
3	53	234721	7	0	0	40	0.0	False	True	False	F
4	28	338409	13	0	0	40	0.0	True	False	False	F

In [40]:
```
x2.drop(['fnlwgt','salary'], axis=1).head()
```

Out[40]:

	age	education-num	capital-gain	capital-loss	hours-per-week	sex_ Female	sex_ Male	race_ Amer-Indian-Eskimo	race_ Asian-Pac-Islander	race_ Black	rac Oth
0	39	13	2174	0	40	False	True	False	False	False	Fal
1	50	13	0	0	13	False	True	False	False	False	Fal
2	38	9	0	0	40	False	True	False	False	False	Fal
3	53	7	0	0	40	False	True	False	False	True	Fal
4	28	13	0	0	40	True	False	False	False	True	Fal

In [41]:
```
model3 = RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [42]:
```
model3.fit(x2.drop(['fnlwgt','salary'], axis=1), x2.salary)
```

Out[42]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [43]:
```
predictions3 = model3.predict(xt2.drop(['fnlwgt','salary'], axis=1))
```

In [44]:
```
list(zip(x2.drop(['fnlwgt','salary'], axis=1).columns, model3.feature_importances_)
```

Out[44]:
```
[('age', 0.20396913845392392),
 ('education-num', 0.21470503836922428),
 ('capital-gain', 0.30169108449458115),
 ('capital-loss', 0.056000789488073015),
 ('hours-per-week', 0.08164056724391218),
 ('sex_ Female', 0.06505710274882708),
 ('sex_ Male', 0.06572254612468864),
 ('race_ Amer-Indian-Eskimo', 0.0005028921287023426),
 ('race_ Asian-Pac-Islander', 0.0003174592158679683),
 ('race_ Black', 0.005200285613206759),
 ('race_ Other', 0.00038261411792322574),
 ('race_ White', 0.004810482001069554)]
```

Classification Report: Race Added

In [45]:
```
print(classification_report(xt2.salary, predictions3))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.97      0.90     12435
         1.0       0.81      0.37      0.51      3846

    accuracy                           0.83     16281
   macro avg       0.82      0.67      0.70     16281
weighted avg       0.83      0.83      0.81     16281
```

Add Marital Status

In [46]:
```
# Columns we want to transform
transform_columns = ['sex', 'race', 'marital-status']

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

In [47]:
```
x2 = df.copy()

x2 = pd.concat([x2.drop(non_num_columns, axis=1),
               pd.get_dummies(df[transform_columns])], axis=1)

x2["salary"] = enc.fit_transform(df[["salary"]])
```

In [48]:
```
xt2 = golden.copy()

xt2 = pd.concat([xt2.drop(non_num_columns, axis=1),
               pd.get_dummies(golden[transform_columns])], axis=1,)

xt2["salary"] = enc.fit_transform(golden[["salary"]])
```

In [49]:
```
model3 = RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [50]:
```
model3.fit(x2.drop(['fnlwgt','salary'], axis=1), x2.salary)
```

Out[50]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [51]:
```
predictions3 = model3.predict(xt2.drop(['fnlwgt','salary'], axis=1))
```

In [52]:
```
list(zip(x2.drop(['fnlwgt','salary'], axis=1).columns, model3.feature_importances_)
```

Out[52]:
```
[('age', 0.10000243541178692),
 ('education-num', 0.15277153412512243),
 ('capital-gain', 0.1999905664848355),
 ('capital-loss', 0.02444892680213238),
 ('hours-per-week', 0.04223911861951419),
 ('sex_ Female', 0.013955158466013957),
 ('sex_ Male', 0.0240204648582359),
 ('race_ Amer-Indian-Eskimo', 0.0002451410021489003),
 ('race_ Asian-Pac-Islander', 0.0002171942592042739),
 ('race_ Black', 0.0020043086125310756),
 ('race_ Other', 0.00026931456629278197),
 ('race_ White', 0.0017297016431381056),
 ('marital-status_ Divorced', 0.0221258603545176 94),
 ('marital-status_ Married-AF-spouse', 0.00030764790626331665),
 ('marital-status_ Married-civ-spouse', 0.2548000424448903),
 ('marital-status_ Married-spouse-absent', 0.00033857938727131873),
 ('marital-status_ Never-married', 0.15507081072225756),
 ('marital-status_ Separated', 0.003359066438246227),
 ('marital-status_ Widowed', 0.002104127895597 1497)]
```

Classification Report: Race + Marital Status Added

In [53]:
```python
print(classification_report(xt2.salary, predictions3))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.96      0.91     12435
         1.0       0.78      0.49      0.61      3846

    accuracy                           0.85     16281
   macro avg       0.82      0.73      0.76     16281
weighted avg       0.84      0.85      0.84     16281
```

Add Education

In [54]:
```python
# Columns we want to transform
transform_columns = ['sex', 'race', 'marital-status', 'education']

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

In [55]:
```python
x2 = df.copy()

x2 = pd.concat([x2.drop(non_num_columns, axis=1),
               pd.get_dummies(df[transform_columns])], axis=1)

x2["salary"] = enc.fit_transform(df[["salary"]])
```

In [56]:
```python
xt2 = golden.copy()

xt2 = pd.concat([xt2.drop(non_num_columns, axis=1),
```

```python
                pd.get_dummies(golden[transform_columns])], axis=1,)

xt2["salary"] = enc.fit_transform(golden[["salary"]])
```

In [57]:
```python
model3.fit(x2.drop(['fnlwgt','salary'], axis=1), x2.salary)
```

Out[57]:
```
RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [58]:
```python
predictions3 = model3.predict(xt2.drop(['fnlwgt','salary'], axis=1))
```

In [59]:
```python
list(zip(x2.drop(['fnlwgt','salary'], axis=1).columns, model3.feature_importances_)
```

Out[59]:
```
[('age', 0.10283087253517507),
 ('education-num', 0.1305042242004478),
 ('capital-gain', 0.1788853819175823),
 ('capital-loss', 0.026788450088421588),
 ('hours-per-week', 0.037984581943269986),
 ('sex_ Female', 0.02832870432369239),
 ('sex_ Male', 0.03160756832559822),
 ('race_ Amer-Indian-Eskimo', 0.00013004149598759676),
 ('race_ Asian-Pac-Islander', 0.00024587006799472506),
 ('race_ Black', 0.001568593956671127),
 ('race_ Other', 0.00022171291983317208),
 ('race_ White', 0.0017271525267160833),
 ('marital-status_ Divorced', 0.015236356427180736),
 ('marital-status_ Married-AF-spouse', 0.00022538485064125461),
 ('marital-status_ Married-civ-spouse', 0.26115335325527406),
 ('marital-status_ Married-spouse-absent', 0.00025257133579515247),
 ('marital-status_ Never-married', 0.11599588150764234),
 ('marital-status_ Separated', 0.0020267044420685114),
 ('marital-status_ Widowed', 0.0006849783725934185),
 ('education_ 10th', 0.0007682611438791228),
 ('education_ 11th', 0.001349874783340316),
 ('education_ 12th', 7.828811283238308e-05),
 ('education_ 1st-4th', 4.794393715196684e-06),
 ('education_ 5th-6th', 0.00047088603318687285),
 ('education_ 7th-8th', 0.0014079422002043016),
 ('education_ 9th', 0.0006671591517304226),
 ('education_ Assoc-acdm', 0.00027943203534194446),
 ('education_ Assoc-voc', 0.00036736426286457154),
 ('education_ Bachelors', 0.01910813724251566),
 ('education_ Doctorate', 0.0058069554874760265),
 ('education_ HS-grad', 0.011226996764468368),
 ('education_ Masters', 0.013712585248383782),
 ('education_ Preschool', 5.9909551287500966e-05),
 ('education_ Prof-school', 0.0072341151271310515),
 ('education_ Some-college', 0.0010589139690569114)]
```

Classification Report: Race + Marital Status + Education Added

In [60]:
```python
print(classification_report(xt2.salary, predictions3))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.96      0.90     12435
         1.0       0.78      0.47      0.59      3846

    accuracy                           0.84     16281
   macro avg       0.82      0.72      0.75     16281
weighted avg       0.84      0.84      0.83     16281
```

Accuracy starting to go back down. I'd assess that we're over-fitting the model now. One more to demonstrate.

Add Workclass

In [61]:
```
# Columns we want to transform
transform_columns = ['sex', 'race', 'marital-status', 'education', 'workclass']

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

In [62]:
```
x2 = df.copy()

x2 = pd.concat([x2.drop(non_num_columns, axis=1),
               pd.get_dummies(df[transform_columns])], axis=1)

x2["salary"] = enc.fit_transform(df[["salary"]])
```

In [63]:
```
xt2 = golden.copy()

xt2 = pd.concat([xt2.drop(non_num_columns, axis=1),
               pd.get_dummies(golden[transform_columns])], axis=1,)

xt2["salary"] = enc.fit_transform(golden[["salary"]])
```

In [64]:
```
model3.fit(x2.drop(['fnlwgt','salary'], axis=1), x2.salary)
```

Out[64]:
```
RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [65]:
```
predictions3 = model3.predict(xt2.drop(['fnlwgt','salary'], axis=1))
```

In [66]:
```
list(zip(x2.drop(['fnlwgt','salary'], axis=1).columns, model3.feature_importances_)
```

Out[66]:
```
[('age', 0.08546377431789369),
 ('education-num', 0.12376036164954692),
 ('capital-gain', 0.1706550422884003),
 ('capital-loss', 0.02939351577096395),
 ('hours-per-week', 0.042738560184155276),
 ('sex_ Female', 0.029081269669669705),
 ('sex_ Male', 0.0299109458259924),
 ('race_ Amer-Indian-Eskimo', 7.141147493838477e-05),
 ('race_ Asian-Pac-Islander', 0.0001329563600748269),
 ('race_ Black', 0.0012846415511346485),
 ('race_ Other', 9.345839633694475e-05),
 ('race_ White', 0.0006085041572384983),
 ('marital-status_ Divorced', 0.025476698865255092),
 ('marital-status_ Married-AF-spouse', 0.00010560628689423182),
 ('marital-status_ Married-civ-spouse', 0.22326868244904635),
 ('marital-status_ Married-spouse-absent', 0.00023973845492389893),
 ('marital-status_ Never-married', 0.1578766472612608),
 ('marital-status_ Separated', 0.0039009376759439576),
 ('marital-status_ Widowed', 0.0023126602036863115),
 ('education_ 10th', 0.0004706196389658293),
 ('education_ 11th', 0.00252202419119483),
 ('education_ 12th', 9.73684010223765e-06),
 ('education_ 1st-4th', 3.8200756682534854e-05),
 ('education_ 5th-6th', 8.468063228392192e-05),
 ('education_ 7th-8th', 0.0016825814809595027),
 ('education_ 9th', 0.000606654486739643),
 ('education_ Assoc-acdm', 6.492860893354904e-05),
 ('education_ Assoc-voc', 0.00011041135168213924),
 ('education_ Bachelors', 0.022352032804729957),
 ('education_ Doctorate', 0.004836755114862793),
 ('education_ HS-grad', 0.008209096893445254),
 ('education_ Masters', 0.013351472324870148),
 ('education_ Preschool', 0.00011072069318974218),
 ('education_ Prof-school', 0.009537548606544429),
 ('education_ Some-college', 0.0016011261372458774),
 ('workclass_ ?', 0.0009849573595121365),
 ('workclass_ Federal-gov', 0.0006892467650210375),
 ('workclass_ Local-gov', 0.00028122883244313126),
 ('workclass_ Never-worked', 0.0),
 ('workclass_ Private', 0.0008775242055209435),
 ('workclass_ Self-emp-inc', 0.0039701506316683535),
 ('workclass_ Self-emp-not-inc', 0.0010377755737777824),
 ('workclass_ State-gov', 0.00019286963101114793),
 ('workclass_ Without-pay', 2.243595256762721e-06)]
```

Classification Report: Race + Marital Status + Education + Workclass Added

In [67]:
```
print(classification_report(xt2.salary, predictions3))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.96      0.90     12435
         1.0       0.79      0.46      0.58      3846

    accuracy                           0.84     16281
   macro avg       0.82      0.71      0.74     16281
weighted avg       0.84      0.84      0.83     16281
```

No change in the classification report with additional column added. We've likely fit the model as best we can, if not overfit. It is possible that we could adjust the columns included to use better data. For example, education and education-num essentially tell us the same thing. We might take back out "education" and include occupation instead.

Remove Education, Add Occupation

In [68]:
```
# Columns we want to transform
transform_columns = ['sex', 'race', 'marital-status', 'occupation', 'workclass']

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

In [69]:
```
x2 = df.copy()

x2 = pd.concat([x2.drop(non_num_columns, axis=1),
               pd.get_dummies(df[transform_columns])], axis=1)

x2["salary"] = enc.fit_transform(df[["salary"]])
```

In [70]:
```
xt2 = golden.copy()

xt2 = pd.concat([xt2.drop(non_num_columns, axis=1),
               pd.get_dummies(golden[transform_columns])], axis=1,)

xt2["salary"] = enc.fit_transform(golden[["salary"]])
```

In [71]:
```
model3.fit(x2.drop(['fnlwgt','salary'], axis=1), x2.salary)
```

Out[71]:
```
RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [72]:
```
predictions3 = model3.predict(xt2.drop(['fnlwgt','salary'], axis=1))
```

In [73]:
```
list(zip(x2.drop(['fnlwgt','salary'], axis=1).columns, model3.feature_importances_)
```

Out[73]:
```
[('age', 0.07516000756742573),
 ('education-num', 0.10447533586660947),
 ('capital-gain', 0.15721873615493048),
 ('capital-loss', 0.022183856371786024),
 ('hours-per-week', 0.044748102495878725),
 ('sex_ Female', 0.03059774425722227),
 ('sex_ Male', 0.03235641369053626),
 ('race_ Amer-Indian-Eskimo', 0.00022708546209541637),
 ('race_ Asian-Pac-Islander', 0.00016767664958095162),
 ('race_ Black', 0.00110353736588602З),
 ('race_ Other', 8.097247720338969e-05),
 ('race_ White', 0.001554956078571229),
 ('marital-status_ Divorced', 0.01706030662835643),
 ('marital-status_ Married-AF-spouse', 0.00014757459683089596),
 ('marital-status_ Married-civ-spouse', 0.2653153961394546),
 ('marital-status_ Married-spouse-absent', 0.0003188602574730129),
 ('marital-status_ Never-married', 0.14863695688882736),
 ('marital-status_ Separated', 0.003776584982626268),
 ('marital-status_ Widowed', 0.0015640236427242767),
 ('occupation_ ?', 0.0016150245495634265),
 ('occupation_ Adm-clerical', 0.003264697088172546),
 ('occupation_ Armed-Forces', 1.496159938838503e-06),
 ('occupation_ Craft-repair', 0.0015886231766470123),
 ('occupation_ Exec-managerial', 0.033231710277816316),
 ('occupation_ Farming-fishing', 0.001133663200268795),
 ('occupation_ Handlers-cleaners', 0.002873178250047366),
 ('occupation_ Machine-op-inspct', 0.0013549676696724482),
 ('occupation_ Other-service', 0.01677776315596376),
 ('occupation_ Priv-house-serv', 5.3278740214637896e-05),
 ('occupation_ Prof-specialty', 0.022247302053842122),
 ('occupation_ Protective-serv', 0.00019242347159751847),
 ('occupation_ Sales', 0.0006481861766207688),
 ('occupation_ Tech-support', 0.0002654507219330733),
 ('occupation_ Transport-moving', 0.0004719583510438247),
 ('workclass_ ?', 0.0008695586923574172),
 ('workclass_ Federal-gov', 0.0004997231952046875),
 ('workclass_ Local-gov', 0.00031480556261776746),
 ('workclass_ Never-worked', 2.564504011761558e-06),
 ('workclass_ Private', 0.0008669189162817387),
 ('workclass_ Self-emp-inc', 0.004480217073137403),
 ('workclass_ Self-emp-not-inc', 0.0004761871344256861),
 ('workclass_ State-gov', 7.539277337729616e-05),
 ('workclass_ Without-pay', 7.815312250663751e-07)]
```

Classification Report: Race + Marital Status + *Occupation* + Workclass Added

In [74]:
```
print(classification_report(xt2.salary, predictions3))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.98      0.90     12435
         1.0       0.86      0.39      0.53      3846

    accuracy                           0.84     16281
   macro avg       0.85      0.68      0.72     16281
weighted avg       0.84      0.84      0.82     16281
```

Same overall accuracy, and relatively comparable precision, recall and f1-score to the previous model.

In [ ]: