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 [278]:
```
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 [279]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
```

In [280]:
```
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [281]:
```
golden.head()
```

Out[281]:

	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 [282]:
```
df.head()
```

Out[282]:

	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	F

In [283]:
```
df.columns
```

Out[283]:
```
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 [284]:
```
from sklearn import preprocessing
```

In [285]:
```
# 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 [286]:
```
dummies = pd.get_dummies(df[transform_columns])
dummies
```

Out[286]:

	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 [287]:
```
dummies.shape
```

Out[287]:
```
(32561, 2)
```

sklearn has a similar process for OneHot Encoding features

In [288]:
```
onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exi
onehot.fit(df[transform_columns])
```

```
/Users/obelisk/anaconda3/lib/python3.10/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[288]:
```
OneHotEncoder(handle_unknown='infrequent_if_exist', sparse=False,
              sparse_output=False)
```

In [289]:
```
onehot.categories_
```

Out[289]:
```
[array([' Female', ' Male'], dtype=object)]
```

In [290]:
```
sex = onehot.transform(df[transform_columns])
sex
```

Out[290]:
```
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       ...,
       [1., 0.],
       [0., 1.],
       [1., 0.]])
```

In [291]:
```
sex.shape
```

Out[291]:
```
(32561, 2)
```

In addition to OneHot encoding there is Ordinal Encoding

In [292]:
```
enc = preprocessing.OrdinalEncoder()
enc.fit(df[["salary"]])
salary = enc.transform(df[["salary"]])
salary
```

Out[292]:
```
array([[0.],
       [0.],
       [0.],
       ...,
       [0.],
       [0.],
       [1.]])
```

In [293]:
```
enc.categories_[0]
```

Out[293]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [294]:
```
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,)


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

```
/Users/obelisk/anaconda3/lib/python3.10/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 [295]:
```
x.head()
```

Out[295]:

	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 [296]:
```
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 [297]:
```
xt.salary.value_counts()
```

Out[297]:
```
0.0    12435
1.0     3846
Name: salary, dtype: int64
```

In [298]:
```
enc.categories_
```

Out[298]:
```
[array([' <=50K.', ' >50K.'], dtype=object)]
```

In [299]:
```
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 [300]:
```
model = RandomForestClassifier(criterion='entropy')
```

In [301]:
```
model = DecisionTreeClassifier(criterion='entropy', max_depth=None)
```

In [302]:
```
model.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)
```

Out[302]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [303]:
```
model.tree_.node_count
```

Out[303]:
```
8321
```

In [304]:
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_im
```

Out[304]:
```
[('age', 0.3248541586650878),
 ('education-num', 0.15995559923545216),
 ('capital-gain', 0.22753078354448814),
 ('capital-loss', 0.07829622512148753),
 ('hours-per-week', 0.1536049769379558),
 (' Female', 0.033776724771279326),
 (' Male', 0.02198153172424917)]
```

In [305]:
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_im
```

Out[305]:
```
[('age', 0.3248541586650878),
 ('education-num', 0.15995559923545216),
 ('capital-gain', 0.22753078354448814),
 ('capital-loss', 0.07829622512148753),
 ('hours-per-week', 0.1536049769379558),
 (' Female', 0.033776724771279326),
 (' Male', 0.02198153172424917)]
```

In [306]:
```
x.drop(['fnlwgt','salary'], axis=1).head()
```

Out[306]:

	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 [307]:
```
set(x.columns) - set(xt.columns)
```

Out[307]:
```
set()
```

In [308]:
```
list(x.drop('salary', axis=1).columns)
```

Out[308]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 ' Female',
 ' Male']
```

In [309]:
```
predictions = model.predict(xt.drop(['fnlwgt','salary'], axis=1))
predictionsx = model.predict(x.drop(['fnlwgt','salary'], axis=1))
```

In [310]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [311]:
```
accuracy_score(xt.salary, predictions)
```

Out[311]:
```
0.8208341010994411
```

In [312]:
```
accuracy_score(xt.salary, predictions)
```

Out[312]:
```
0.8208341010994411
```

In [313]:
```
confusion_matrix(xt.salary, predictions)
```

Out[313]:
```
array([[11456,   979],
       [ 1938,  1908]])
```

In [314]:
```
print(classification_report(xt.salary, predictions))
```

```
              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 [315]:
```
print(classification_report(xt.salary, predictions))
```

```
              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 [316]:
```
accuracy_score(x.salary, predictionsx)
```

Out[316]:
```
0.8955806025613464
```

In [317]:
```
confusion_matrix(x.salary, predictionsx)
```

Out[317]:
```
array([[24097,   623],
       [ 2777,  5064]])
```

In [318]:
```
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 [319]:
```
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 [320]:
```
# Defining the Models and setting depth
deci_tree= DecisionTreeClassifier(criterion='entropy', max_depth=38)
rand_fore = RandomForestClassifier(criterion='entropy', max_depth=38)
```

In [321]:
```
# Fitting the Models
deci_tree.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)
rand_fore.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[321]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=38)
```

In [322]:
```
# Predicting
pred_dt = deci_tree.predict(xt.drop(['fnlwgt','salary'], axis=1))
pred_dtx = deci_tree.predict(x.drop(['fnlwgt','salary'], axis=1))
pred_rf = rand_fore.predict(xt.drop(['fnlwgt','salary'], axis=1))
pred_rfx = rand_fore.predict(x.drop(['fnlwgt','salary'], axis=1))
```

In [323]:
```
# Decision Tree Accuracy Score (because I am curious)
accuracy_score(xt.salary, pred_dt)
```

Out[323]:
```
0.8213254714083902
```

In [324]:
```
# Random Forest Accuracy Score (because I am curious)
accuracy_score(xt.salary, pred_rf)
```

Out[324]:
```
0.8273447576930164
```

In [325]:
```
# Decision Tree Classification Report
print(classification_report(xt.salary, pred_dt))
```

```
              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 [326]:
```
# Random Forest Classification Report
print(classification_report(xt.salary, pred_rf))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89     12435
         1.0       0.67      0.52      0.59      3846

    accuracy                           0.83     16281
   macro avg       0.77      0.72      0.74     16281
weighted avg       0.82      0.83      0.82     16281
```

In [327]:
```
# Decision Tree Confusion Matrix
confusion_matrix(xt.salary, pred_dt)
```

Out[327]:
```
array([[11473,   962],
       [ 1947,  1899]])
```

In [328]:
```
# Random Forest Confusion Matrix
confusion_matrix(xt.salary, pred_rf)
```

Out[328]:
```
array([[11471,   964],
       [ 1847,  1999]])
```

I tested a fair number of fixed depths. Generally speaking at the two extremes, a shallow max depth of <5, or a extremely deep max depth of >100, both models performed about the same. Between 6 and 50 Random Forest showed an average improvement in accuracy by about 1%, with slightly better overall precision, between 1% and 5%, and slightly better recall, also between 1% and 5%. Overall both models tend to struggle predicting salaries greather than $50k based on the information available.

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 [329]:
```
# Defining Basline Model
rand_fore = RandomForestClassifier(criterion='entropy', max_depth=None
```

In [330]:
```
# Fitting Baseline Model
rand_fore.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[330]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [331]:
```
# Predicting Baseline Model
pred_rf = rand_fore.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [332]:
```
# Random Forest Classification Report for reference
print(classification_report(xt.salary, pred_rf))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89     12435
         1.0       0.68      0.52      0.59      3846

    accuracy                           0.83     16281
   macro avg       0.77      0.72      0.74     16281
weighted avg       0.82      0.83      0.82     16281
```

In [333]:
```
# Random Forest Confusion Matrix for reference
confusion_matrix(xt.salary, pred_rf)
```

Out[333]:
```
array([[11482,   953],
       [ 1847,  1999]])
```

In [334]:
```
# Here for Quick reference of what columns I need to add.
#non_num_columns = [, , ,
                    #, , , ,
                    #]
```

In [335]:
```
# Processing 'race' column and testing
enc.fit(df[['race']])
race = enc.transform(df[['race']])
race
```

Out[335]:
```
array([[4.],
       [4.],
       [4.],
       ...,
       [4.],
       [4.],
       [4.]])
```

In [336]:
```
# Adding 'race' to x
enc.fit(df[['race']])
x['race'] = enc.transform(df[['race']])
x.head()
```

Out[336]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	Female	Male	race
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	4.0
2	38	215646	9	0	0	40	0.0	0.0	1.0	4.0
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 [337]:
```
# Adding 'race' to xt
enc.fit(golden[['race']])
xt['race'] = enc.transform(golden[['race']])
```

In [338]:
```
# Defining Model 1
rand_fore1 = RandomForestClassifier(criterion='entropy', max_depth=Non
```

In [339]:
```
# Fitting Model 1
rand_fore1.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[339]:
```
▾          RandomForestClassifier

RandomForestClassifier(criterion='entropy')
```

In [340]:
```
# Predicting Model 1
pred_rf1 = rand_fore1.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [341]:
```
# Random Forest 1 Classification Report
print(classification_report(xt.salary, pred_rf1))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89     12435
         1.0       0.67      0.51      0.58      3846

    accuracy                           0.83     16281
   macro avg       0.77      0.72      0.74     16281
weighted avg       0.82      0.83      0.82     16281
```

In [342]:
```
# Random Forest 1 Confusion Matrix
confusion_matrix(xt.salary, pred_rf1)
```

Out[342]:
```
array([[11472,   963],
       [ 1876,  1970]])
```

In [343]:
```
# Processing 'workclass' column and testing
enc.fit(df[['workclass']])
workclass = enc.transform(df[['workclass']])
workclass
```

Out[343]:
```
array([[7.],
       [6.],
       [4.],
       ...,
       [4.],
       [4.],
       [5.]])
```

In [344]:
```
# Adding 'workclass' to x
enc.fit(df[['workclass']])
x['workclass'] = enc.transform(df[['workclass']])
x.head()
```

Out[344]:

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

In [345]:
```
# Adding 'workclass' to xt
enc.fit(golden[['workclass']])
xt['workclass'] = enc.transform(golden[['workclass']])
```

In [346]:
```
# Defining Model 2
rand_fore2 = RandomForestClassifier(criterion='entropy', max_depth=Non
```

In [347]:
```
# Fitting Model 2
rand_fore2.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[347]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [348]:
```
# Predicting Model 2
pred_rf2 = rand_fore2.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [349]:
```
# Random Forest 2 Classification Report
print(classification_report(xt.salary, pred_rf2))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89     12435
         1.0       0.68      0.53      0.59      3846

    accuracy                           0.83     16281
   macro avg       0.77      0.72      0.74     16281
weighted avg       0.82      0.83      0.82     16281
```

In [350]:
```
# Random Forest 2 Confusion Matrix
confusion_matrix(xt.salary, pred_rf2)
```

Out[350]:
```
array([[11462,   973],
       [ 1825,  2021]])
```

In [351]:
```
# Processing 'education' column and testing
enc.fit(df[['education']])
education = enc.transform(df[['education']])
education
```

Out[351]:
```
array([[ 9.],
       [ 9.],
       [11.],
       ...,
       [11.],
       [11.],
       [11.]])
```

In [352]:
```
# Adding 'education' to x
enc.fit(df[['education']])
x['education'] = enc.transform(df[['education']])
x.head()
```

Out[352]:

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

In [353]:
```
# Adding 'education' to xt
enc.fit(golden[['education']])
xt['education'] = enc.transform(golden[['education']])
```

In [354]:
```
# Defining Model 3
rand_fore3 = RandomForestClassifier(criterion='entropy', max_depth=Non
```

In [355]:
```
# Fitting Model 3
rand_fore3.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[355]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [356]:
```
# Predicting Model 3
pred_rf3 = rand_fore3.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [357]:
```
# Random Forest 3 Classification Report
print(classification_report(xt.salary, pred_rf3))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89     12435
         1.0       0.67      0.52      0.59      3846

    accuracy                           0.83     16281
   macro avg       0.77      0.72      0.74     16281
weighted avg       0.82      0.83      0.82     16281
```

In [358]:
```
# Random Forest 3 Confusion Matrix
confusion_matrix(xt.salary, pred_rf3)
```

Out[358]:
```
array([[11452,   983],
       [ 1831,  2015]])
```

In [359]:
```
# Processing 'marital-status' column and testing
enc.fit(df[['marital-status']])
marital_status = enc.transform(df[['marital-status']])
marital_status
```

Out[359]:
```
array([[4.],
       [2.],
       [0.],
       ...,
       [6.],
       [4.],
       [2.]])
```

In [360]:
```
# Adding 'marital-status' to x
enc.fit(df[['marital-status']])
x['marital-status'] = enc.transform(df[['marital-status']])
x.head()
```

Out[360]:

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

In [361]:
```
# Adding 'marital-status' to xt
enc.fit(golden[['marital-status']])
xt['marital-status'] = enc.transform(golden[['marital-status']])
```

In [362]:
```
# Defining Model 4
rand_fore4 = RandomForestClassifier(criterion='entropy', max_depth=Non
```

In [363]:
```
# Fitting Model 4
rand_fore4.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[363]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [364]:
```
# Predicting Model 4
pred_rf4 = rand_fore4.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [365]:
```
# Random Forest 4 Classification Report, probably the best model so fa
print(classification_report(xt.salary, pred_rf4))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.92      0.90     12435
         1.0       0.70      0.60      0.65      3846

    accuracy                           0.84     16281
   macro avg       0.79      0.76      0.77     16281
weighted avg       0.84      0.84      0.84     16281
```

In [366]:
```
# Random Forest 4 Confusion Matrix
confusion_matrix(xt.salary, pred_rf4)
```

Out[366]:
```
array([[11422,  1013],
       [ 1527,  2319]])
```

In [367]:
```
# Processing 'occupation' column and testing
enc.fit(df[['occupation']])
occupation = enc.transform(df[['occupation']])
occupation
```

Out[367]:
```
array([[1.],
       [4.],
       [6.],
       ...,
       [1.],
       [1.],
       [4.]])
```

In [368]:
```
# Adding 'occupation' to x
enc.fit(df[['occupation']])
x['occupation'] = enc.transform(df[['occupation']])
x.head()
```

Out[368]:

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

In [369]:
```
# Adding 'occupation' to xt
enc.fit(golden[['occupation']])
xt['occupation'] = enc.transform(golden[['occupation']])
```

In [370]:
```
# Defining Model 5
rand_fore5 = RandomForestClassifier(criterion='entropy', max_depth=Non
```

In [371]:
```
# Fitting Model 5
rand_fore5.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[371]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [372]:
```
# Predicting Model 5
pred_rf5 = rand_fore5.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [373]:
```
# Random Forest 5 Classification Report
print(classification_report(xt.salary, pred_rf5))
```

```
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.90     12435
         1.0       0.71      0.62      0.66      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.77      0.78     16281
weighted avg       0.84      0.85      0.84     16281
```

In [374]:
```
# Random Forest 5 Confusion Matrix
confusion_matrix(xt.salary, pred_rf5)
```

Out[374]:
```
array([[11446,   989],
       [ 1479,  2367]])
```

In [375]:
```
# Processing 'native-country' column and testing
enc.fit(df[['native-country']])
native_country = enc.transform(df[['native-country']])
native_country
```

Out[375]:
```
array([[39.],
       [39.],
       [39.],
       ...,
       [39.],
       [39.],
       [39.]])
```

In [376]:
```
# Adding 'native-country' to x
enc.fit(df[['native-country']])
x['native-country'] = enc.transform(df[['native-country']])
x.head()
```

Out[376]:

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

In [377]:
```
# Adding 'native-country' to xt
enc.fit(golden[['native-country']])
xt['native-country'] = enc.transform(golden[['native-country']])
```

In [378]:
```
# Defining Model 6
rand_fore6 = RandomForestClassifier(criterion='entropy', max_depth=Non
```

In [379]:
```
# Fitting Model 6
rand_fore6.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[379]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [380]:
```
# Predicting Model 6
pred_rf6 = rand_fore6.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [381]:
```
# Random Forest 6 Classification Report
print(classification_report(xt.salary, pred_rf6))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.92      0.90     12435
         1.0       0.71      0.60      0.65      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.76      0.78     16281
weighted avg       0.84      0.85      0.84     16281
```

In [382]:
```
# Random Forest 6 Confusion Matrix
confusion_matrix(xt.salary, pred_rf6)
```

Out[382]:
```
array([[11487,   948],
       [ 1522,  2324]])
```

In [383]:
```
# Processing 'relationship' column and testing
enc.fit(df[['relationship']])
relationship = enc.transform(df[['relationship']])
relationship
```

Out[383]:
```
array([[1.],
       [0.],
       [1.],
       ...,
       [4.],
       [3.],
       [5.]])
```

In [384]:
```
# Adding 'relationship' to x
enc.fit(df[['relationship']])
x['relationship'] = enc.transform(df[['relationship']])
x.head()
```

Out[384]:

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

In [385]:
```
# Adding 'relationship' to xt
enc.fit(golden[['relationship']])
xt['relationship'] = enc.transform(golden[['relationship']])
```

In [386]:
```
# Defining Model 7
rand_fore7 = RandomForestClassifier(criterion='entropy', max_depth=Non
```

In [387]:
```
# Fitting Model 7
rand_fore7.fit(x.drop(['fnlwgt', 'salary'], axis=1), x.salary)
```

Out[387]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [388]:
```
# Predicting Model 7
pred_rf7 = rand_fore7.predict(xt.drop(['fnlwgt','salary'], axis=1))
```

In [389]:
```
# Random Forest 7 Classification Report
print(classification_report(xt.salary, pred_rf7))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.93      0.90     12435
         1.0       0.71      0.60      0.65      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.76      0.78     16281
weighted avg       0.84      0.85      0.84     16281
```

In [390]:
```
# Random Forest 7 Confusion Matrix
confusion_matrix(xt.salary, pred_rf7)
```

Out[390]:
```
array([[11512,   923],
       [ 1536,  2310]])
```