# Assignment is below at the end

- 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.plot_tree.html

In [325…

```
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 [326…

```
df = pd.read_csv('adult.data', index_col=False)
```

In [327…

```
golden = pd.read_csv('adult.test', index_col=False)
```

In [328…

```
golden.head()
```

Out[328]:

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

In [329…

```
df.head()
```

Out[329]:

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

In [330…

```
df.columns
```

Out[330]:

```
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 [331…

```
from sklearn import preprocessing
```

In [332…

```
# 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 [333…

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

Out[333]:

| | 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 [334…

```
dummies.shape
```

Out[334]:

```
(32561, 2)
```

# sklearn has a similar process for OneHot Encoding features

In [335…

```
onehot = preprocessing.OneHotEncoder(handle_unknown = "infrequent_if_exist", sparse=Fa
onehot.fit(df[transform_columns])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remov
ed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

Out[335]:

```
▾                          OneHotEncoder
OneHotEncoder(handle_unknown='infrequent_if_exist', sparse=False,
              sparse_output=False)
```

In [336…

```
onehot.categories_
```

Out[336]:

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

In [337…

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

```
Out[337]: array([[0., 1.],
                 [0., 1.],
                 [0., 1.],
                 ...,
                 [1., 0.],
                 [0., 1.],
                 [1., 0.]])
```

```
In [338…  sex.shape
```

```
Out[338]: (32561, 2)
```

# In addition to OneHot encoding there is Ordinal Encoding

```
In [339…  enc = preprocessing.OrdinalEncoder()
          enc.fit(df[["salary"]])
          salary = enc.transform(df[["salary"]])
          salary
```

```
Out[339]: array([[0.],
                 [0.],
                 [0.],
                 ...,
                 [0.],
                 [0.],
                 [1.]])
```

```
In [340…  enc.categories_[0]
```

```
Out[340]: array([' <=50K', ' >50K'], dtype=object)
```

```
In [341…  x = df.copy()

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


          onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fals

          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"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: FutureWarning: `sparse` was renamed to `sparse_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 [342…

```
x
```

Out[342]:

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **32556** | 27 | 257302 | 12 | 0 | 0 | 38 | 0.0 | 1.0 | 0.0 |
| **32557** | 40 | 154374 | 9 | 0 | 0 | 40 | 1.0 | 0.0 | 1.0 |
| **32558** | 58 | 151910 | 9 | 0 | 0 | 40 | 0.0 | 1.0 | 0.0 |
| **32559** | 22 | 201490 | 9 | 0 | 0 | 20 | 0.0 | 0.0 | 1.0 |
| **32560** | 52 | 287927 | 9 | 15024 | 0 | 40 | 1.0 | 1.0 | 0.0 |

32561 rows × 9 columns

In [343…

```
xt = golden.copy()

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

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

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

In [344…

```
xt.salary.value_counts()
```

Out[344]:

```
0.0    12435
1.0     3846
Name: salary, dtype: int64
```

In [345…

```
enc.categories_
```

Out[345]:

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

In [346…

```python
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 [347…

```python
model = RandomForestClassifier(criterion='entropy')
```

In [348…

```python
model = DecisionTreeClassifier(criterion='entropy', max_depth=None)
```

In [349…

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

Out[349]:

```
▾          DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [350…

```python
model.tree_.node_count
```

Out[350]:

```
8313
```

In [351…

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

Out[351]:

```
[('age', 0.32449288950363436),
 ('education-num', 0.16041765339065917),
 ('capital-gain', 0.22731243895177794),
 ('capital-loss', 0.07830610027050318),
 ('hours-per-week', 0.15395455662211974),
 (' Female', 0.03396418514452267),
 (' Male', 0.021552176116782933)]
```

In [352…

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

Out[352]:

```
[('age', 0.32449288950363436),
 ('education-num', 0.16041765339065917),
 ('capital-gain', 0.22731243895177794),
 ('capital-loss', 0.07830610027050318),
 ('hours-per-week', 0.15395455662211974),
 (' Female', 0.03396418514452267),
 (' Male', 0.021552176116782933)]
```

In [353…

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

Out[353]:

| | 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 [354…

```python
set(x.columns) - set(xt.columns)
```

```
Out[354]: set()
```

```
In [355…  list(x.drop('salary', axis=1).columns)
```

```
Out[355]: ['age',
           'fnlwgt',
           'education-num',
           'capital-gain',
           'capital-loss',
           'hours-per-week',
           ' Female',
           ' Male']
```

```
In [356…  predictions = model.predict(xt.drop(['fnlwgt','salary'], axis=1))
          predictionsx = model.predict(x.drop(['fnlwgt','salary'], axis=1))
```

```
In [357…  from sklearn.metrics import (
              accuracy_score,
              classification_report,
              confusion_matrix, auc, roc_curve
          )
```

```
In [358…  accuracy_score(xt.salary, predictions)
```

```
Out[358]: 0.8208341010994411
```

```
In [359…  confusion_matrix(xt.salary, predictions)
```

```
Out[359]: array([[11460,   975],
                 [ 1942,  1904]], dtype=int64)
```

```
In [360…  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 [361…  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 [362…  accuracy_score(x.salary, predictionsx)
```

```
Out[362]: 0.8955806025613464
```

In [363…

```
confusion_matrix(x.salary, predictionsx)
```

Out[363]:

```
array([[24097,   623],
       [ 2777,  5064]], dtype=int64)
```

In [364…

```
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 [365…

```
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 [366…

```
modela = RandomForestClassifier(criterion = 'entropy', max_depth = 5)
modelb = DecisionTreeClassifier(criterion = 'entropy', max_depth = 5)
```

In [367…

```
modela.fit(x.drop(['fnlwgt','salary'], axis = 1), x.salary)
modelb.fit(x.drop(['fnlwgt','salary'], axis = 1), x.salary)
```

Out[367]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [368…

```
predictionsa = modela.predict(xt.drop(['fnlwgt','salary'], axis = 1))
predictionsax = modela.predict(x.drop(['fnlwgt','salary'], axis = 1))
predictionsb = modelb.predict(xt.drop(['fnlwgt','salary'], axis = 1))
predictionsbx = modelb.predict(x.drop(['fnlwgt','salary'], axis = 1))
```

In [369…

```
accuracy_score(xt.salary, predictionsa)
```

Out[369]:

```
0.8312142988759904
```

In [370…

```
accuracy_score(xt.salary, predictionsb)
```

Out[370]:

```
0.8201584669246361
```

In [371…

```
confusion_matrix(xt.salary, predictionsa)
```

Out[371]:

```
array([[12049,   386],
       [ 2362,  1484]], dtype=int64)
```

In [372…

```
confusion_matrix(xt.salary, predictionsb)
```

Out[372]:

```
array([[11458,   977],
       [ 1951,  1895]], dtype=int64)
```

In [373…

```
print(classification_report(xt.salary, predictionsa))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.97      0.90     12435
         1.0       0.79      0.39      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 [374…

```
print(classification_report(xt.salary, predictionsb))
```

```
              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
```

# 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 [375…

```
##Workclass
transform_columns1 = ['workclass']
non_num_columns1 = ['education', 'marital-status',
                    'occupation', 'relationship', 'race', 'sex',
                    'native-country']
```

In [397…

```
data1 = df.copy()

onehot1 = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fal

enc1 = preprocessing.OrdinalEncoder()

enc1.fit(df[["workclass"]])


transformed1 = onehot1.transform(df[transform_columns1])
new_cols1 = list(onehot1.categories_[0].flatten())
df_trans1 = pd.DataFrame(transformed1, columns = new_cols1)


data1 = pd.concat(
    [
        data1.drop(non_num_columns1, axis = 1),
        df_trans,
        df_trans1
    ],
    axis = 1,)


data1["workclass"] = enc1.fit_transform(df[["workclass"]])
data1["sex"] = enc1.fit_transform(df[["sex"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remov
ed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [377…

```
data1x = golden.copy()

transformed1 = onehot1.transform(golden[transform_columns1])
new_cols1 = list(onehot1.categories_[0].flatten())
df_trans1x = pd.DataFrame(transformed1, columns = new_cols1)

data1x = pd.concat(
    [
        data1x.drop(non_num_columns1, axis = 1),
        df_transx,
        df_trans1x
    ],
    axis = 1,)

data1x["workclass"] = enc1.fit_transform(golden[["workclass"]])
data1x["sex"] = enc1.fit_transform(golden[["sex"]])
```

In [378…

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

In [379…

```
modelc.fit(data1.drop(['fnlwgt','salary'], axis = 1), data1.salary)
```

Out[379]:

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

In [380…

```
predictionsc = modelc.predict(data1x.drop(['fnlwgt','salary'], axis = 1))
predictionscx = modelc.predict(data1.drop(['fnlwgt','salary'], axis = 1))
```

In [381…

```
confusion_matrix(data1.salary, predictionscx)
```

Out[381]:

```
array([[24196,   524],
       [ 4879,  2962]], dtype=int64)
```

In [382…

```
print(classification_report(data1.salary, predictionscx))
```

```
              precision    recall  f1-score   support

       <=50K       0.83      0.98      0.90     24720
        >50K       0.85      0.38      0.52      7841

    accuracy                           0.83     32561
   macro avg       0.84      0.68      0.71     32561
weighted avg       0.84      0.83      0.81     32561
```

In [435…

```
##Martial Status
transform_columns2 = ['marital-status']
non_num_columns2 = ['education', 'occupation', 'relationship', 'race', 'sex', 'native-
```

In [426…

```
data2 = df.copy()

onehot2 = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fal

enc2 = preprocessing.OrdinalEncoder()

enc2.fit(df[["marital-status"]])


transformed2 = onehot2.transform(df[transform_columns2])
new_cols2 = list(onehot2.categories_[0].flatten())
df_trans2 = pd.DataFrame(transformed2, columns = new_cols2)


data2 = pd.concat(
    [
        data2.drop(non_num_columns2, axis = 1),
        df_trans,
        df_trans1,
        df_trans2
    ],
    axis = 1,)

data2["workclass"] = enc2.fit_transform(df[["workclass"]])
data2["marital-status"] = enc2.fit_transform(df[["marital-status"]])
data2["sex"] = enc2.fit_transform(df[["sex"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: FutureWarning: `sparse` was renamed to `sparse_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 [431…

```
data2x = golden.copy()

transformed2 = onehot2.transform(golden[transform_columns2])
new_cols2 = list(onehot2.categories_[0].flatten())
df_trans2x = pd.DataFrame(transformed2, columns = new_cols2)

data2x = pd.concat(
    [
        data2x.drop(non_num_columns2, axis = 1),
        df_transx,
        df_trans1x,
        df_trans2x
    ],
    axis = 1,)

data2x["workclass"] = enc2.fit_transform(golden[["workclass"]])
data2x["marital-status"] = enc2.fit_transform(golden[["marital-status"]])
data2x["sex"] = enc2.fit_transform(golden[["sex"]])
```

In [440…

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

In [441…

```
modeld.fit(data2.drop(['fnlwgt','salary'], axis = 1), data2.salary)
```

Out[441]:

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

In [442…

```
predictionsd = modeld.predict(data2x.drop(['fnlwgt','salary'], axis = 1))
predictionsdx = modeld.predict(data2.drop(['fnlwgt','salary'], axis = 1))
```

In [443…

```
confusion_matrix(data2.salary, predictionsdx)
```

Out[443]:

```
array([[23688,  1032],
       [ 3788,  4053]], dtype=int64)
```

In [444…

```
print(classification_report(data2.salary, predictionsdx))
```

```
              precision    recall  f1-score   support

       <=50K       0.86      0.96      0.91     24720
        >50K       0.80      0.52      0.63      7841

    accuracy                           0.85     32561
   macro avg       0.83      0.74      0.77     32561
weighted avg       0.85      0.85      0.84     32561
```

In [436...

```
##Education
transform_columns3 = ['education']
non_num_columns3 = ['occupation', 'relationship', 'race', 'sex', 'native-country']
```

In [437...

```
data3 = df.copy()

onehot3 = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fal

enc3 = preprocessing.OrdinalEncoder()

enc3.fit(df[["education"]])


transformed3 = onehot3.transform(df[transform_columns3])
new_cols3 = list(onehot3.categories_[0].flatten())
df_trans3 = pd.DataFrame(transformed3, columns = new_cols3)


data3 = pd.concat(
    [
        data3.drop(non_num_columns3, axis = 1),
        df_trans,
        df_trans1,
        df_trans2,
        df_trans3
    ],
    axis = 1,)

data3["workclass"] = enc3.fit_transform(df[["workclass"]])
data3["marital-status"] = enc3.fit_transform(df[["marital-status"]])
data3["education"] = enc3.fit_transform(df[["education"]])
data3["sex"] = enc3.fit_transform(df[["sex"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remov
ed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [439...

```
data3x = golden.copy()

transformed3 = onehot3.transform(golden[transform_columns3])
new_cols3 = list(onehot3.categories_[0].flatten())
df_trans3x = pd.DataFrame(transformed3, columns = new_cols3)

data3x = pd.concat(
    [
        data3x.drop(non_num_columns3, axis = 1),
        df_transx,
        df_trans1x,
        df_trans2x,
        df_trans3x
    ],
    axis = 1,)

data3x["workclass"] = enc3.fit_transform(golden[["workclass"]])
data3x["marital-status"] = enc3.fit_transform(golden[["marital-status"]])
data3x["education"] = enc3.fit_transform(golden[["education"]])
data3x["sex"] = enc3.fit_transform(golden[["sex"]])
```

In [445…

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

In [446…

```
modele.fit(data3.drop(['fnlwgt','salary'], axis = 1), data3.salary)
```

Out[446]:

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

In [447…

```
predictionse = modele.predict(data3x.drop(['fnlwgt','salary'], axis = 1))
predictionsex = modele.predict(data3.drop(['fnlwgt','salary'], axis = 1))
```

In [448…

```
confusion_matrix(data3.salary, predictionsex)
```

Out[448]:

```
array([[23642,  1078],
       [ 4005,  3836]], dtype=int64)
```

In [449…

```
print(classification_report(data3.salary, predictionsex))
```

```
              precision    recall  f1-score   support

       <=50K       0.86      0.96      0.90     24720
        >50K       0.78      0.49      0.60      7841

    accuracy                           0.84     32561
   macro avg       0.82      0.72      0.75     32561
weighted avg       0.84      0.84      0.83     32561
```

In [460…

```
##Occupation
transform_columns4 = ['occupation']
non_num_columns4 = ['relationship', 'race', 'sex', 'native-country']
```

In [451…

```
data4 = df.copy()

onehot4 = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fal

enc4 = preprocessing.OrdinalEncoder()

enc4.fit(df[["occupation"]])


transformed4 = onehot4.transform(df[transform_columns4])
new_cols4 = list(onehot4.categories_[0].flatten())
df_trans4 = pd.DataFrame(transformed4, columns = new_cols4)


data4 = pd.concat(
    [
        data4.drop(non_num_columns4, axis = 1),
        df_trans,
        df_trans1,
        df_trans2,
        df_trans3,
        df_trans4
    ],
    axis = 1,)
```

```
data4["workclass"] = enc4.fit_transform(df[["workclass"]])
data4["marital-status"] = enc4.fit_transform(df[["marital-status"]])
data4["education"] = enc4.fit_transform(df[["education"]])
data4["occupation"] = enc4.fit_transform(df[["occupation"]])
data4["sex"] = enc4.fit_transform(df[["sex"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remov
ed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [452…

```
data4x = golden.copy()

transformed4 = onehot4.transform(golden[transform_columns4])
new_cols4 = list(onehot4.categories_[0].flatten())
df_trans4x = pd.DataFrame(transformed4, columns = new_cols4)

data4x = pd.concat(
    [
        data4x.drop(non_num_columns4, axis = 1),
        df_transx,
        df_trans1x,
        df_trans2x,
        df_trans3x,
        df_trans4x
    ],
    axis = 1,)

data4x["workclass"] = enc4.fit_transform(golden[["workclass"]])
data4x["marital-status"] = enc4.fit_transform(golden[["marital-status"]])
data4x["education"] = enc4.fit_transform(golden[["education"]])
data4x["occupation"] = enc4.fit_transform(golden[["occupation"]])
data4x["sex"] = enc4.fit_transform(golden[["sex"]])
```

In [453…

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

In [454…

```
modelf.fit(data4.drop(['fnlwgt','salary'], axis = 1), data4.salary)
```

Out[454]:

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

In [455…

```
predictionsf = modelf.predict(data4x.drop(['fnlwgt','salary'], axis = 1))
predictionsfx = modelf.predict(data4.drop(['fnlwgt','salary'], axis = 1))
```

In [456…

```
confusion_matrix(data4.salary, predictionsfx)
```

Out[456]:

```
array([[24005,   715],
       [ 4329,  3512]], dtype=int64)
```

In [457…

```
print(classification_report(data4.salary, predictionsfx))
```

```
              precision    recall  f1-score   support

        <=50K       0.85      0.97      0.90     24720
         >50K       0.83      0.45      0.58      7841

     accuracy                           0.85     32561
    macro avg       0.84      0.71      0.74     32561
 weighted avg       0.84      0.85      0.83     32561
```

In [470…

```
##Relationship
transform_columns5 = ['relationship']
non_num_columns5 = ['race', 'sex', 'native-country']
```

In [461…

```
data5 = df.copy()

onehot5 = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fal

enc5 = preprocessing.OrdinalEncoder()

enc5.fit(df[["relationship"]])


transformed5 = onehot5.transform(df[transform_columns5])
new_cols5 = list(onehot5.categories_[0].flatten())
df_trans5 = pd.DataFrame(transformed5, columns = new_cols5)


data5 = pd.concat(
    [
        data5.drop(non_num_columns5, axis = 1),
        df_trans,
        df_trans1,
        df_trans2,
        df_trans3,
        df_trans4,
        df_trans5
    ],
    axis = 1,)

data5["workclass"] = enc5.fit_transform(df[["workclass"]])
data5["marital-status"] = enc5.fit_transform(df[["marital-status"]])
data5["education"] = enc5.fit_transform(df[["education"]])
data5["occupation"] = enc5.fit_transform(df[["occupation"]])
data5["relationship"] = enc5.fit_transform(df[["relationship"]])
data5["sex"] = enc5.fit_transform(df[["sex"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remov
ed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [462…

```
data5x = golden.copy()

transformed5 = onehot5.transform(golden[transform_columns5])
new_cols5 = list(onehot5.categories_[0].flatten())
df_trans5x = pd.DataFrame(transformed5, columns = new_cols5)

data5x = pd.concat(
    [
        data5x.drop(non_num_columns5, axis = 1),
        df_transx,
        df_trans1x,
        df_trans2x,
        df_trans3x,
        df_trans4x,
        df_trans5x
    ],
    axis = 1,)

data5x["workclass"] = enc5.fit_transform(golden[["workclass"]])
data5x["marital-status"] = enc5.fit_transform(golden[["marital-status"]])
data5x["education"] = enc5.fit_transform(golden[["education"]])
data5x["occupation"] = enc5.fit_transform(golden[["occupation"]])
data5x["relationship"] = enc5.fit_transform(golden[["relationship"]])
data5x["sex"] = enc5.fit_transform(golden[["sex"]])
```

In [463…

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

In [464…

```
modelg.fit(data5.drop(['fnlwgt','salary'], axis = 1), data5.salary)
```

Out[464]:

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

In [465…

```
predictionsg = modelg.predict(data5x.drop(['fnlwgt','salary'], axis = 1))
predictionsgx = modelg.predict(data5.drop(['fnlwgt','salary'], axis = 1))
```

In [466…

```
confusion_matrix(data5.salary, predictionsgx)
```

Out[466]:

```
array([[23697,  1023],
       [ 4016,  3825]], dtype=int64)
```

In [467…

```
print(classification_report(data5.salary, predictionsgx))
```

```
              precision    recall  f1-score   support

       <=50K       0.86      0.96      0.90     24720
        >50K       0.79      0.49      0.60      7841

    accuracy                           0.85     32561
   macro avg       0.82      0.72      0.75     32561
weighted avg       0.84      0.85      0.83     32561
```

In [481…

```
##Race
transform_columns6 = ['race']
non_num_columns6 = ['native-country']
```

In [486…

```
data6 = df.copy()

onehot6 = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fal

enc6 = preprocessing.OrdinalEncoder()

enc6.fit(df[["race"]])
```

```
transformed6 = onehot6.transform(df[transform_columns6])
new_cols6 = list(onehot6.categories_[0].flatten())
df_trans6 = pd.DataFrame(transformed6, columns = new_cols6)


data6 = pd.concat(
    [
        data6.drop(non_num_columns6, axis = 1),
        df_trans,
        df_trans1,
        df_trans2,
        df_trans3,
        df_trans4,
        df_trans5,
        df_trans6
    ],
    axis = 1,)

data6["workclass"] = enc6.fit_transform(df[["workclass"]])
data6["marital-status"] = enc6.fit_transform(df[["marital-status"]])
data6["education"] = enc6.fit_transform(df[["education"]])
data6["occupation"] = enc6.fit_transform(df[["occupation"]])
data6["relationship"] = enc6.fit_transform(df[["relationship"]])
data6["race"] = enc6.fit_transform(df[["race"]])
data6["sex"] = enc6.fit_transform(df[["sex"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remov
ed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [487…

```
data6x = golden.copy()

transformed6 = onehot6.transform(golden[transform_columns6])
new_cols6 = list(onehot6.categories_[0].flatten())
df_trans6x = pd.DataFrame(transformed6, columns = new_cols6)

data6x = pd.concat(
    [
        data6x.drop(non_num_columns6, axis = 1),
        df_transx,
        df_trans1x,
        df_trans2x,
        df_trans3x,
        df_trans4x,
        df_trans5x,
        df_trans6x
    ],
    axis = 1,)

data6x["workclass"] = enc6.fit_transform(golden[["workclass"]])
data6x["marital-status"] = enc6.fit_transform(golden[["marital-status"]])
data6x["education"] = enc6.fit_transform(golden[["education"]])
data6x["occupation"] = enc6.fit_transform(golden[["occupation"]])
data6x["relationship"] = enc6.fit_transform(golden[["relationship"]])
data6x["race"] = enc6.fit_transform(golden[["race"]])
data6x["sex"] = enc6.fit_transform(golden[["sex"]])
```

In [488…

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

In [489…

```
modelh.fit(data6.drop(['fnlwgt','salary'], axis = 1), data6.salary)
```

Out[489]:

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

In [490…

```
predictionsh = modelh.predict(data6x.drop(['fnlwgt','salary'], axis = 1))
predictionshx = modelh.predict(data6.drop(['fnlwgt','salary'], axis = 1))
```

In [491…

```
confusion_matrix(data6.salary, predictionshx)
```

Out[491]:

```
array([[23668,  1052],
       [ 4027,  3814]], dtype=int64)
```

In [492…

```
print(classification_report(data6.salary, predictionshx))
```

```
              precision    recall  f1-score   support

       <=50K       0.85      0.96      0.90     24720
        >50K       0.78      0.49      0.60      7841

    accuracy                           0.84     32561
   macro avg       0.82      0.72      0.75     32561
weighted avg       0.84      0.84      0.83     32561
```

In [493…

```
##Native Country
transform_columns7 = ['native-country']
```

In [495…

```
data7 = df.copy()

onehot7 = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=Fal

enc7 = preprocessing.OrdinalEncoder()

enc7.fit(df[["native-country"]])


transformed7 = onehot7.transform(df[transform_columns7])
new_cols7 = list(onehot7.categories_[0].flatten())
df_trans7 = pd.DataFrame(transformed7, columns = new_cols7)


data7 = pd.concat(
    [
        data7,
        df_trans,
        df_trans1,
        df_trans2,
        df_trans3,
        df_trans4,
        df_trans5,
        df_trans6,
        df_trans7
    ],
    axis = 1,)

data7["workclass"] = enc7.fit_transform(df[["workclass"]])
data7["marital-status"] = enc7.fit_transform(df[["marital-status"]])
data7["education"] = enc7.fit_transform(df[["education"]])
data7["occupation"] = enc7.fit_transform(df[["occupation"]])
data7["relationship"] = enc7.fit_transform(df[["relationship"]])
data7["race"] = enc7.fit_transform(df[["race"]])
data7["sex"] = enc7.fit_transform(df[["sex"]])
data7["native-country"] = enc7.fit_transform(df[["native-country"]])
```

```
C:\Users\jorda\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:828: Fu
tureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be remov
ed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [496…

```
data7x = golden.copy()

transformed7 = onehot7.transform(golden[transform_columns7])
new_cols7 = list(onehot7.categories_[0].flatten())
df_trans7x = pd.DataFrame(transformed7, columns = new_cols7)

data7x = pd.concat(
    [
        data7x,
        df_transx,
        df_trans1x,
        df_trans2x,
        df_trans3x,
        df_trans4x,
        df_trans5x,
        df_trans6x,
        df_trans7x
    ],
    axis = 1,)

data7x["workclass"] = enc7.fit_transform(golden[["workclass"]])
data7x["marital-status"] = enc7.fit_transform(golden[["marital-status"]])
data7x["education"] = enc7.fit_transform(golden[["education"]])
data7x["occupation"] = enc7.fit_transform(golden[["occupation"]])
data7x["relationship"] = enc7.fit_transform(golden[["relationship"]])
data7x["race"] = enc7.fit_transform(golden[["race"]])
data7x["sex"] = enc7.fit_transform(golden[["sex"]])
data7x["native-country"] = enc7.fit_transform(golden[["native-country"]])
```

In [497…

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

In [498…

```
modeli.fit(data7.drop(['fnlwgt','salary'], axis = 1), data7.salary)
```

Out[498]:

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

In [500…

```
predictionsi = modeli.predict(data7x.drop(['fnlwgt','salary'], axis = 1))
predictionsix = modeli.predict(data7.drop(['fnlwgt','salary'], axis = 1))
```

In [501…

```
confusion_matrix(data7.salary, predictionsix)
```

Out[501]:

```
array([[23641,  1079],
       [ 4210,  3631]], dtype=int64)
```

In [502…

```
print(classification_report(data7.salary, predictionsix))
```

```
              precision    recall  f1-score   support

       <=50K       0.85      0.96      0.90     24720
        >50K       0.77      0.46      0.58      7841

    accuracy                           0.84     32561
   macro avg       0.81      0.71      0.74     32561
weighted avg       0.83      0.84      0.82     32561
```

In [ ]:

```

```