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 [1]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (20, 6)
plt.rcParams['font.size'] = 14
import pandas as pd
```

In [2]:
```python
df = pd.read_csv('../data/adult.data', index_col=False)
```

In [3]:
```python
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [4]:
```python
golden.head()
```

Out[4]:

	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 [5]:
```python
df.head()
```

Out[5]:

	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 [6]:
```python
df.columns
```

Out[6]:
```
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 [7]:
```python
from sklearn import preprocessing
```

In [8]:
```python
# 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 [9]:
```python
dummies = pd.get_dummies(df[transform_columns])
dummies
```

Out[9]:

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

32561 rows × 2 columns

In [10]:
```python
dummies.shape
```

Out[10]:
```
(32561, 2)
```

sklearn has a similar process for OneHot Encoding features

In [11]:
```python
onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=False)
onehot.fit(df[transform_columns])
```

```
/Users/kevinbrogan/anaconda3/lib/python3.10/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(
```

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

In [12]:
```python
onehot.categories_
```

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

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

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

In [14]:
```python
sex.shape
```

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

In addition to OneHot encoding there is Ordinal Encoding

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

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

In [16]:
```python
enc.categories_[0]
```

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

In [17]:
```python
x = df.copy()

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


onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=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"]])
```

```
/Users/kevinbrogan/anaconda3/lib/python3.10/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 [18]:
```python
x.head()
```

Out[18]:

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

In [19]:
```python
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 [20]:
```python
xt.salary.value_counts()
```

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

In [21]:
```python
enc.categories_
```

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

In [22]:
```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 [23]:
```python
model = RandomForestClassifier(criterion='entropy')
```

In [24]:
```python
model = DecisionTreeClassifier(criterion='entropy', max_depth=None)
```

In [25]:
```python
model.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)
```

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

In [26]:
```python
model.tree_.node_count
```

Out[26]:
```
8343
```

In [27]:
```python
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```

Out[27]:
```
[('age', 0.3224696797613446),
 ('education-num', 0.16039663827087378),
 ('capital-gain', 0.22779810532320635),
 ('capital-loss', 0.07851624804931023),
 ('hours-per-week', 0.1555059695817829),
 (' Female', 0.001074996873666915),
 (' Male', 0.054238362139815366)]
```

In [28]:
```python
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```

Out[28]:
```
[('age', 0.3224696797613446),
 ('education-num', 0.16039663827087378),
 ('capital-gain', 0.22779810532320635),
 ('capital-loss', 0.07851624804931023),
 ('hours-per-week', 0.1555059695817829),
 (' Female', 0.001074996873666915),
 (' Male', 0.054238362139815366)]
```

In [29]:
```python
x.drop(['fnlwgt','salary'], axis=1).head()
```

Out[29]:

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

In [30]:
```python
set(x.columns) - set(xt.columns)
```

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

In [31]:
```python
list(x.drop('salary', axis=1).columns)
```

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

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

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

In [34]:
```python
accuracy_score(xt.salary, predictions)
```

Out[34]:
```
0.8217554204287206
```

In [35]:
```python
accuracy_score(xt.salary, predictions)
```

Out[35]:
```
0.8217554204287206
```

In [36]:
```python
confusion_matrix(xt.salary, predictions)
```

Out[36]:
```
array([[11465,   970],
       [ 1932,  1914]])
```

In [37]:
```python
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 [38]:
```python
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 [39]:
```python
accuracy_score(x.salary, predictionsx)
```

Out[39]:
```
0.8955806025613464
```

In [40]:
```python
confusion_matrix(x.salary, predictionsx)
```

Out[40]:
```
array([[24097,   623],
       [ 2777,  5064]])
```

In [41]:
```python
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 [42]:
```python
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)

Note: I am using the datasets that were created in the previous steps (called "x" and "xt") as the training set. I make a few edits to the dataset to fit the parameters of the assignment question, but will using the x and golden datasets, above, as my base.

In [43]:
```python
x.head()
# xt.head()
```

Out[43]:

	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 [44]:
```python
x = x.drop([' Female', ' Male', 'fnlwgt'], axis=1)
xt = xt.drop([' Female', ' Male', 'fnlwgt'], axis=1)
```

In [45]:
```python
x.head()
# xt.head()
```

Out[45]:

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

In [46]:
```python
modelDTC = DecisionTreeClassifier(criterion='entropy', max_depth=3)
modelRFC = RandomForestClassifier(criterion='entropy')
```

In [47]:
```python
modelDTC.fit(x.drop(['salary'], axis=1), x.salary)
modelRFC.fit(x.drop(['salary'], axis=1), x.salary)
```

Out[47]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [48]:
```python
modelDTC.tree_.node_count
```

Out[48]:
```
15
```

In [49]:
```python
predictions_DTC = modelDTC.predict(xt.drop(['salary'], axis=1))
# predictions_xDTC = modelDTC.predict(x.drop(['salary'], axis=1))
```

In [50]:
```python
predictions_RFC = modelRFC.predict(xt.drop(['salary'], axis=1))
# predictions_xRFC = modelRFC.predict(x.drop(['salary'], axis=1))
```

In [51]:
```python
print(confusion_matrix(xt.salary, predictions_DTC))
```

```
[[12428     7]
 [ 3198   648]]
```

In [52]:
```python
confusion_matrix(xt.salary, predictions_RFC)
```

Out[52]:
```
array([[11570,   865],
       [ 2005,  1841]])
```

In [53]:
```python
print(classification_report(xt.salary, predictions_DTC))
```

```
              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
```

In [54]:
```python
print(classification_report(xt.salary, predictions_RFC))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.93      0.89     12435
         1.0       0.68      0.48      0.56      3846

    accuracy                           0.82     16281
   macro avg       0.77      0.70      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.

First Iteration, marital-status

In [55]:
```python
transform_cols = ['marital-status']
```

In [56]:
```python
onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=False).fit(df[transform_cols])
```

```
/Users/kevinbrogan/anaconda3/lib/python3.10/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 [57]:
```python
onehot.categories_
```

Out[57]:
```
[array([' Divorced', ' Married-AF-spouse', ' Married-civ-spouse',
        ' Married-spouse-absent', ' Never-married', ' Separated',
        ' Widowed'], dtype=object)]
```

In [58]:
```python
transformed = onehot.transform(df[transform_cols])
new_cols_marstat = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols_marstat)

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

In [59]:
```python
transformed = onehot.transform(golden[transform_cols])
new_cols_marstat = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols_marstat)

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

In [60]:
```python
modelRFC_2 = RandomForestClassifier(criterion='entropy')
```

In [61]:
```python
modelRFC_2.fit(x.drop(['salary'], axis=1), x.salary)
```

Out[61]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [62]:
```python
predictions = modelRFC_2.predict(xt.drop(['salary'], axis=1))
```

In [63]:
```python
confusion_matrix(xt.salary, predictions)
```

Out[63]:
```
array([[11408,  1027],
       [ 1495,  2351]])
```

In [64]:
```python
print(classification_report(xt.salary, predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.92      0.90     12435
         1.0       0.70      0.61      0.65      3846

    accuracy                           0.85     16281
   macro avg       0.79      0.76      0.78     16281
weighted avg       0.84      0.85      0.84     16281
```

Second Iteration, marital-status + race

In [65]:
```python
transform_cols = ['race']
onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=False).fit(df[transform_cols])
```

```
/Users/kevinbrogan/anaconda3/lib/python3.10/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 [66]:
```python
onehot.categories_
```

Out[66]:
```
[array([' Amer-Indian-Eskimo', ' Asian-Pac-Islander', ' Black', ' Other',
        ' White'], dtype=object)]
```

In [67]:
```python
transformed = onehot.transform(df[transform_cols])
new_cols_race = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols_race)

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

In [68]:
```python
transformed = onehot.transform(df[transform_cols])
new_cols_race = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols_race)

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

In [69]:
```python
modelRFC_3 = RandomForestClassifier(criterion='entropy')
modelRFC_3.fit(x.drop(['salary'], axis=1), x.salary)
```

Out[69]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [70]:
```python
predictions_iter2 = modelRFC_3.predict(xt.drop(['salary'], axis=1))
```

In [71]:
```python
confusion_matrix(xt.salary, predictions_iter2)
```

Out[71]:
```
array([[11402,  1033],
       [ 1541,  2305]])
```

In [72]:
```python
print(classification_report(xt.salary, predictions_iter2))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.92      0.90     12435
         1.0       0.69      0.60      0.64      3846

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

Third Iteration, marital-status + race + workclass

In [73]:
```python
transform_cols = ['workclass']
onehot = preprocessing.OneHotEncoder(handle_unknown="infrequent_if_exist", sparse=False).fit(df[transform_cols])
```

```
/Users/kevinbrogan/anaconda3/lib/python3.10/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 [74]:
```python
onehot.categories_
```

Out[74]:
```
[array([' ?', ' Federal-gov', ' Local-gov', ' Never-worked', ' Private',
        ' Self-emp-inc', ' Self-emp-not-inc', ' State-gov', ' Without-pay'],
       dtype=object)]
```

In [75]:
```python
transformed = onehot.transform(df[transform_cols])
new_cols_class = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols_class)

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

In [76]:
```python
transformed = onehot.transform(df[transform_cols])
new_cols_class = list(onehot.categories_[0].flatten())
df_trans = pd.DataFrame(transformed, columns=new_cols_class)

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

In [77]:
```python
modelRFC_4 = RandomForestClassifier(criterion='entropy')
modelRFC_4.fit(x.drop(['salary'], axis=1), x.salary)
```

Out[77]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [78]:
```python
predictions_iter3 = modelRFC_4.predict(xt.drop(['salary'], axis=1))
```

In [79]:
```python
confusion_matrix(xt.salary, predictions_iter3)
```

Out[79]:
```
array([[11340,  1095],
       [ 1569,  2277]])
```

In [80]:
```python
print(classification_report(xt.salary, predictions_iter3))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.91      0.89     12435
         1.0       0.68      0.59      0.63      3846

    accuracy                           0.84     16281
   macro avg       0.78      0.75      0.76     16281
weighted avg       0.83      0.84      0.83     16281
```

In [ ]: