Assignment_Week6

In [1]:
```
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]:
```
df = pd.read_csv('adult.data', index_col=False)
```

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

In [4]:
```
from sklearn import preprocessing
```

In [45]:
```
# 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 [6]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

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)

1-A. preprocessing data

In [7]:
```
enc = preprocessing.OrdinalEncoder()
```

In [30]:
```
#Using x as datatable
x = df.copy()

#Drop non-numerical variables
x = x.drop(non_num_columns, axis=1)

#Transfrom salary column into ordinal variable
enc.fit(df[["salary"]])
x["salary"] = enc.transform(df[["salary"]])

x.head()
```

Out[30]:

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

In [31]:
```
#Do the same process to test set

xt = golden.copy()
xt = xt.drop(non_num_columns, axis=1)

enc.fit(golden[["salary"]])
xt["salary"] = enc.transform(golden[["salary"]])

xt.head()
```

Out[31]:

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

1-B. Random Forest

In [10]:
```
#Define 'fmodel' for RandomForestClassifier model
fmodel = RandomForestClassifier(criterion='entropy')

#Fit the model
fmodel.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)

#Make prediction values
predictionsf = fmodel.predict(xt.drop(['fnlwgt','salary'], axis=1))

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

```
[('age', 0.34795420669088023), ('education-num', 0.1743008357929181), ('capital-gain', 0.20925554651330278),
('capital-loss', 0.0824119676028227), ('hours-per-week', 0.18607744340007623)]
```

In [11]:
```
#Print Classification Report and Confusion matrix
print(classification_report(xt.salary, predictionsf))
print(confusion_matrix(xt.salary, predictionsf))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.93      0.89     12435
         1.0       0.69      0.47      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

[[11609   826]
 [ 2027  1819]]
```

1-C. Decision Tree

In [29]:
```
#Define 'tmodel' for DecisionTreeClassifier model
tmodel = DecisionTreeClassifier(criterion='entropy', max_depth=10)

#Fit the model
tmodel.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)

#Make prediction values
predictionst = tmodel.predict(xt.drop(['fnlwgt','salary'], axis=1))

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

```
[('age', 0.2797023323111284), ('education-num', 0.19619282670936367), ('capital-gain', 0.31836074204340603),
('capital-loss', 0.12655536766564787), ('hours-per-week', 0.07918873127045413)]
```

1-C-1. Classification report for Max_depth=None

Result: RandomForest outperforms DecisionTree(precision, recall, f1-score are same or higher)

In [24]:
```
#DecisionTree(Max_depth=None)
print(classification_report(xt.salary, predictionst))
print(confusion_matrix(xt.salary, predictionst))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.93      0.89     12435
         1.0       0.66      0.46      0.54      3846

    accuracy                           0.82     16281
   macro avg       0.75      0.69      0.71     16281
weighted avg       0.80      0.82      0.80     16281

[[11514   921]
 [ 2067  1779]]
```

In [22]:
```
#RandomForest
print(classification_report(xt.salary, predictionsf))
print(confusion_matrix(xt.salary, predictionsf))
```

```
              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      0.71      0.73     16281
weighted avg       0.81      0.82      0.81     16281

[[11570   865]
 [ 1996  1850]]
```

1-C-2. Classification report for Max_depth=5

Result: f1 score is higher in RandomForest, but some precision and recall scores are higher in DecisionTree

In [27]:
```
#DecisionTree (Max_depth=5)
print(classification_report(xt.salary, predictionst))
print(confusion_matrix(xt.salary, predictionst))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.95      0.89     12435
         1.0       0.74      0.42      0.53      3846

    accuracy                           0.83     16281
   macro avg       0.79      0.68      0.71     16281
weighted avg       0.82      0.83      0.81     16281

[[11862   573]
 [ 2246  1600]]
```

In [32]:
```
#RandomForest
print(classification_report(xt.salary, predictionsf))
print(confusion_matrix(xt.salary, predictionsf))
```

```
              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      0.71      0.73     16281
weighted avg       0.81      0.82      0.81     16281

[[11570   865]
 [ 1996  1850]]
```

1-C-3. Classification report for Max_depth=10

Result: DecisionTree outperforms RandomForest

In [31]:
```
# DecisionTree
print(classification_report(xt.salary, predictionst))
print(confusion_matrix(xt.salary, predictionst))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.94      0.90     12435
         1.0       0.72      0.48      0.58      3846

    accuracy                           0.83     16281
   macro avg       0.79      0.71      0.74     16281
weighted avg       0.82      0.83      0.82     16281

[[11704   731]
 [ 1992  1854]]
```

In [33]:
```
#RandomForest
print(classification_report(xt.salary, predictionsf))
print(confusion_matrix(xt.salary, predictionsf))
```

```
              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      0.71      0.73     16281
weighted avg       0.81      0.82      0.81     16281

[[11570   865]
 [ 1996  1850]]
```

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.

2-A. Add column "workclass" to train and test set

In [26]:
```
#Using OrdincalEncoder to convert non-numerical values into numerical values.
enc = preprocessing.OrdinalEncoder()
enc.fit(df[["workclass"]])
x2 = x.copy()
x2["workclass"] = enc.fit_transform(df[["workclass"]])
x2.head()
```

Out[26]:

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

In [27]:
```
enc.fit(golden[["workclass"]])
xt2 = xt.copy()
xt2["workclass"] = enc.fit_transform(golden[["workclass"]])
xt2.head()
```

Out[27]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	workclass
0	25	226802	7	0	0	40	0.0	4.0
1	38	89814	9	0	0	50	0.0	4.0
2	28	336951	12	0	0	40	1.0	2.0
3	44	160323	10	7688	0	40	1.0	4.0
4	18	103497	10	0	0	30	0.0	0.0

In [54]:
```
#Define 'tmodel2' for DecisionTreeClassifier model
tmodel2 = DecisionTreeClassifier(criterion='entropy', max_depth=None)

#Fit the model
tmodel2.fit(x2.drop(['fnlwgt','salary'], axis=1), x2.salary)

#Make prediction values
predictionst2 = tmodel2.predict(xt2.drop(['fnlwgt','salary'], axis=1))

print(list(zip(x2.drop(['fnlwgt','salary'], axis=1).columns, tmodel2.feature_importances_)))
```

```
[('age', 0.32808031922556546), ('education-num', 0.1540474036969254), ('capital-gain', 0.21237296825153737),
('capital-loss', 0.0815816523808854), ('hours-per-week', 0.14710798304077805), ('workclass', 0.0768096734043
0823)]
```

In [55]:
```
# Classification Report and Confusion Matrix
print(classification_report(xt2.salary, predictionst2))
print(confusion_matrix(xt2.salary, predictionst2))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.91      0.88     12435
         1.0       0.63      0.47      0.54      3846

    accuracy                           0.81     16281
   macro avg       0.74      0.69      0.71     16281
weighted avg       0.80      0.81      0.80     16281

[[11361  1074]
 [ 2020  1826]]
```

Result : f1-score worsen (0.89 -> 0.88, 0.54 -> 0.54)

2-B. Add column "education" to train and test set

In [34]:
```
enc.fit(df[["education"]])
x3 = x.copy()
x3["education"] = enc.fit_transform(df[["education"]])
x3.head()
```

Out[34]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	education
0	39	77516	13	2174	0	40	0.0	9.0
1	50	83311	13	0	0	13	0.0	9.0
2	38	215646	9	0	0	40	0.0	11.0
3	53	234721	7	0	0	40	0.0	1.0
4	28	338409	13	0	0	40	0.0	9.0

In [36]:
```
enc.fit(golden[["education"]])
xt3 = xt.copy()
xt3["education"] = enc.fit_transform(golden[["education"]])
xt3.head()
```

Out[36]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	education
0	25	226802	7	0	0	40	0.0	1.0
1	38	89814	9	0	0	50	0.0	11.0
2	28	336951	12	0	0	40	1.0	7.0
3	44	160323	10	7688	0	40	1.0	15.0
4	18	103497	10	0	0	30	0.0	15.0

In [56]:
```
#Define 'tmodel3' for DecisionTreeClassifier model
tmodel3 = DecisionTreeClassifier(criterion='entropy', max_depth=None)

#Fit the model
tmodel3.fit(x3.drop(['fnlwgt','salary'], axis=1), x3.salary)

#Make prediction values
predictionst3 = tmodel3.predict(xt3.drop(['fnlwgt','salary'], axis=1))

print(list(zip(x3.drop(['fnlwgt','salary'], axis=1).columns, tmodel3.feature_importances_)))
```

```
[('age', 0.3266372524993378), ('education-num', 0.1459110448847573), ('capital-gain', 0.24856422340651133),
('capital-loss', 0.09769279930756909), ('hours-per-week', 0.16094251130990372), ('education', 0.020252168591
920593)]
```

In [57]:
```
# Classification Report and Confusion Matrix
print(classification_report(xt3.salary, predictionst3))
print(confusion_matrix(xt3.salary, predictionst3))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.93      0.89     12435
         1.0       0.66      0.46      0.54      3846

    accuracy                           0.82     16281
   macro avg       0.75      0.69      0.72     16281
weighted avg       0.80      0.82      0.81     16281

[[11524   911]
 [ 2065  1781]]
```

Result : same f1-score (0.89 -> 0.89, 0.54 -> 0.54)

2-C. Add all non-numerical columns at once to train and test set

In [43]:
```
x4 = x.copy()

enc.fit(df[["workclass"]])
x4["workclass"] = enc.fit_transform(df[["workclass"]])

enc.fit(df[["education"]])
x4["education"] = enc.fit_transform(df[["education"]])

enc.fit(df[["marital-status"]])
x4["marital-status"] = enc.fit_transform(df[["marital-status"]])

enc.fit(df[["occupation"]])
x4["occupation"] = enc.fit_transform(df[["occupation"]])

enc.fit(df[["relationship"]])
x4["relationship"] = enc.fit_transform(df[["relationship"]])

enc.fit(df[["race"]])
x4["race"] = enc.fit_transform(df[["race"]])

enc.fit(df[["sex"]])
x4["sex"] = enc.fit_transform(df[["sex"]])

enc.fit(df[["native-country"]])
x4["native-country"] = enc.fit_transform(df[["native-country"]])

x4.head()
```

Out[43]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	workclass	education	marital-status	occupation	relationship	race
0	39	77516	13	2174	0	40	0.0	7.0	9.0	4.0	1.0	1.0	4.0
1	50	83311	13	0	0	13	0.0	6.0	9.0	2.0	4.0	0.0	4.0
2	38	215646	9	0	0	40	0.0	4.0	11.0	0.0	6.0	1.0	4.0
3	53	234721	7	0	0	40	0.0	4.0	1.0	2.0	6.0	0.0	2.0
4	28	338409	13	0	0	40	0.0	4.0	9.0	2.0	10.0	5.0	2.0

In [58]:
```
xt4 = xt.copy()

enc.fit(golden[["workclass"]])
xt4["workclass"] = enc.fit_transform(golden[["workclass"]])

enc.fit(golden[["education"]])
xt4["education"] = enc.fit_transform(golden[["education"]])

enc.fit(golden[["marital-status"]])
xt4["marital-status"] = enc.fit_transform(golden[["marital-status"]])

enc.fit(golden[["occupation"]])
xt4["occupation"] = enc.fit_transform(golden[["occupation"]])

enc.fit(golden[["relationship"]])
xt4["relationship"] = enc.fit_transform(golden[["relationship"]])

enc.fit(golden[["race"]])
xt4["race"] = enc.fit_transform(golden[["race"]])

enc.fit(golden[["sex"]])
xt4["sex"] = enc.fit_transform(golden[["sex"]])

enc.fit(golden[["native-country"]])
xt4["native-country"] = enc.fit_transform(golden[["native-country"]])

xt4.head()
```

Out[58]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	workclass	education	marital-status	occupation	relationship	race
0	25	226802	7	0	0	40	0.0	4.0	1.0	4.0	7.0	3.0	2.0
1	38	89814	9	0	0	50	0.0	4.0	11.0	2.0	5.0	0.0	4.0
2	28	336951	12	0	0	40	1.0	2.0	7.0	2.0	11.0	0.0	4.0
3	44	160323	10	7688	0	40	1.0	4.0	15.0	2.0	7.0	0.0	2.0
4	18	103497	10	0	0	30	0.0	0.0	15.0	4.0	0.0	3.0	4.0

In [60]:
```
#Define 'tmodel4' for DecisionTreeClassifier model
tmodel4 = DecisionTreeClassifier(criterion='entropy', max_depth=None)

#Fit the model
tmodel4.fit(x4.drop(['fnlwgt','salary'], axis=1), x4.salary)

#Make prediction values
predictionst4 = tmodel4.predict(xt4.drop(['fnlwgt','salary'], axis=1))

print(list(zip(x4.drop(['fnlwgt','salary'], axis=1).columns, tmodel4.feature_importances_)))
```

```
[('age', 0.19116634444695357), ('education-num', 0.11507685916151264), ('capital-gain', 0.11844622187936933), ('capital-loss', 0.04035827014613208), ('hours-per-week', 0.10945879312003691), ('workclass', 0.049382852822795406), ('education', 0.01829320361440402), ('marital-status', 0.01524543880525042), ('occupation', 0.08163331557431314), ('relationship', 0.213666359821733), ('race', 0.02041834268497339), ('sex', 0.004820611966873391), ('native-country', 0.022033385955652785)]
```

In [74]:
```
# Classification Report and Confusion Matrix
print(classification_report(xt4.salary, predictionst4))
print(confusion_matrix(xt4.salary, predictionst4))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.88      0.88     12435
         1.0       0.62      0.60      0.61      3846

    accuracy                           0.82     16281
   macro avg       0.75      0.74      0.75     16281
weighted avg       0.82      0.82      0.82     16281

[[11000  1435]
 [ 1525  2321]]
```

Result : changed f1-score (0.89 -> 0.88, 0.54 -> 0.61) and found out that 'relationship' has high importance in predicting 'salary'

2-D. Add column "relationship" to train and test set

In [64]:
```
#Using OrdincalEncoder to convert non-numerical values into numerical values.
enc = preprocessing.OrdinalEncoder()
enc.fit(df[["relationship"]])
x5 = x.copy()
x5["relationship"] = enc.fit_transform(df[["relationship"]])
x5.head()
```

Out[64]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	relationship
0	39	77516	13	2174	0	40	0.0	1.0
1	50	83311	13	0	0	13	0.0	0.0
2	38	215646	9	0	0	40	0.0	1.0
3	53	234721	7	0	0	40	0.0	0.0
4	28	338409	13	0	0	40	0.0	5.0

In [69]:
```
#Using OrdincalEncoder to convert non-numerical values into numerical values.
enc.fit(golden[["relationship"]])
xt5 = x.copy()
xt5["relationship"] = enc.fit_transform(df[["relationship"]])
xt5.head()
```

Out[69]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary	relationship
0	39	77516	13	2174	0	40	0.0	1.0
1	50	83311	13	0	0	13	0.0	0.0
2	38	215646	9	0	0	40	0.0	1.0
3	53	234721	7	0	0	40	0.0	0.0
4	28	338409	13	0	0	40	0.0	5.0

In [71]:
```
#Define 'tmodel5' for DecisionTreeClassifier model
tmodel5 = DecisionTreeClassifier(criterion='entropy', max_depth=None)

#Fit the model
tmodel5.fit(x5.drop(['fnlwgt','salary'], axis=1), x5.salary)

#Make prediction values
predictionst5 = tmodel5.predict(xt5.drop(['fnlwgt','salary'], axis=1))

print(list(zip(x5.drop(['fnlwgt','salary'], axis=1).columns, tmodel5.feature_importances_)))
```

```
[('age', 0.20165691621485987), ('education-num', 0.16082291455446976), ('capital-gain', 0.1595678003108816
4), ('capital-loss', 0.05716584141169078), ('hours-per-week', 0.13991633840981874), ('relationship', 0.28087
018909827927)]
```

In [76]:
```
# Classification Report and Confusion Matrix
print(classification_report(xt5.salary, predictionst5))
print(confusion_matrix(xt5.salary, predictionst5))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.97      0.95     24720
         1.0       0.90      0.74      0.81      7841

    accuracy                           0.92     32561
   macro avg       0.91      0.86      0.88     32561
weighted avg       0.92      0.92      0.91     32561

[[24055   665]
 [ 2053  5788]]
```

Result : significantly improved f1-score (0.89 -> 0.95, 0.54 -> 0.81)

Overall Result of Assignment 2 : the input value 'relationship' had high importance in predicting salary. Therefore, the model improved by taking the input value 'relationship' into account.

Adding high-relevant input variable into model would improve performance; adding low-relevant input variable may not improve or harm performance