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 [1]:
```
# Import Settings
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]:
```
# Import Classifiers
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
# Import OneHotEncoder (To Simplify Process)
from sklearn.preprocessing import OneHotEncoder
```

In [3]:
```
# Import Forest and Tree Models
forest_model = RandomForestClassifier(criterion='entropy', max_depth=5)
tree_model = DecisionTreeClassifier(criterion='entropy', max_depth=5) # Max Depth is s
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [4]:
```
# Import Data
df = pd.read_csv('../data/adult.data', index_col=False)
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [5]:
```
df_num = df[['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']
golden_num = golden[['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-pe
df_num
golden_num
```

Out[5]:

	age	education-num	capital-gain	capital-loss	hours-per-week
0	25	7	0	0	40
1	38	9	0	0	50
2	28	12	0	0	40
3	44	10	7688	0	40
4	18	10	0	0	30
...	...	...	...	...	...
16276	39	13	0	0	36
16277	64	9	0	0	40
16278	38	13	0	0	50
16279	44	13	5455	0	40
16280	35	13	0	0	60

16281 rows × 5 columns

In [6]:
```
def salary_over_under(s):
    if s == ' <=50K.':
        return ' <=50K'
    elif s == ' >50K.':
        return ' >50K'
```

In [7]:
```
# Encoding
onehot_encoder = OneHotEncoder(handle_unknown='ignore')
onehot_encoder.fit(df[['salary']])
```

Out[7]:
```
OneHotEncoder(handle_unknown='ignore')
```

In [8]:
```
df_salary = onehot_encoder.fit_transform(df[['salary']]).toarray()
df_salary = pd.DataFrame(df_salary, columns = onehot_encoder.categories_[0])
golden_salary = onehot_encoder.fit_transform(golden[['salary']]).toarray()
golden_salary = pd.DataFrame(golden_salary, columns = onehot_encoder.categories_[0])
df_salary_y = df_salary[[' >50K']]
golden_salary_y = golden_salary[[' >50K.']]
```

In [9]:
```
# Constructing forest model off of DF
forest_model.fit(df_num, df_salary_y)
```

```
C:\Users\Brett\AppData\Local\Temp\ipykernel_26048\329273109.py:2: DataConversionWarni
ng: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples,), for example using ravel().
  forest_model.fit(df_num, df_salary_y)
```

Out[9]:
```
RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [10]:
```
# Find Predictions from forest model
forest_model_predictions = forest_model.predict(golden_num)
```

In [11]:
```
# Precision, Recall, and F1,
print(classification_report(golden_salary_y, forest_model_predictions))
# Confusion Matrix
print(confusion_matrix(golden_salary_y, forest_model_predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.97      0.90     12435
         1.0       0.79      0.39      0.53      3846

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

[[12028   407]
 [ 2332  1514]]
```

In [12]:
```
# Constructing tree model off of DF
tree_model.fit(df_num, df_salary_y)
```

Out[12]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [13]:
```
# Find predictions from tree model
decision_tree_predictions = tree_model.predict(golden_num)
```

In [14]:
```
# Precision, Recall, and F1,
print(classification_report(golden_salary_y, decision_tree_predictions))
# Confusion Matrix
print(confusion_matrix(golden_salary_y, decision_tree_predictions))
```

```
              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      0.69      0.71     16281
weighted avg       0.82      0.83      0.81     16281

[[11861   574]
 [ 2245  1601]]
```

In [15]:
```
# For a max depth of 5, the model's performance is mostly similar; however, the Random
# identifying positives (12028 vs 11861).
```

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 [16]:
```
# Encoding Workclass
## df
onehot_encoder = OneHotEncoder(handle_unknown='ignore')
onehot_encoder.fit(df[['workclass']])
df_new_workclass = onehot_encoder.fit_transform(df[['workclass']]).toarray()
df_new_workclass = pd.DataFrame(df_new_workclass, columns = onehot_encoder.categories_
df_new_workclass

## golden
onehot_encoder = OneHotEncoder(handle_unknown='ignore')
onehot_encoder.fit(golden[['workclass']])
golden_new_workclass = onehot_encoder.fit_transform(golden[['workclass']]).toarray()
golden_new_workclass = pd.DataFrame(golden_new_workclass, columns = onehot_encoder.cat
golden_new_workclass
```

Out[16]:

	?	Federal-gov	Local-gov	Never-worked	Private	Self-emp-inc	Self-emp-not-inc	State-gov	Without-pay
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
16276	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
16277	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16278	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
16279	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
16280	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

16281 rows × 9 columns

In [17]:
```
# Join to Original
# DF
df_num
df_num_workclass = df_num.join(df_new_workclass)
df_num_workclass
# Golden
golden_num
```

```
golden_num_workclass = golden_num.join(golden_new_workclass)
golden_num_workclass
```

Out[17]:

	age	education-num	capital-gain	capital-loss	hours-per-week	?	Federal-gov	Local-gov	Never-worked	Private	Self-emp-inc	Self emp not in
0	25	7	0	0	40	0.0	0.0	0.0	0.0	1.0	0.0	0.
1	38	9	0	0	50	0.0	0.0	0.0	0.0	1.0	0.0	0.
2	28	12	0	0	40	0.0	0.0	1.0	0.0	0.0	0.0	0.
3	44	10	7688	0	40	0.0	0.0	0.0	0.0	1.0	0.0	0.
4	18	10	0	0	30	1.0	0.0	0.0	0.0	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...
16276	39	13	0	0	36	0.0	0.0	0.0	0.0	1.0	0.0	0.
16277	64	9	0	0	40	1.0	0.0	0.0	0.0	0.0	0.0	0.
16278	38	13	0	0	50	0.0	0.0	0.0	0.0	1.0	0.0	0.
16279	44	13	5455	0	40	0.0	0.0	0.0	0.0	1.0	0.0	0.
16280	35	13	0	0	60	0.0	0.0	0.0	0.0	0.0	1.0	0.

16281 rows × 14 columns

In [18]:
```
# Constructing tree model off of new DF with education
forest_model.fit(df_num_workclass, df_salary_y)
```

```
C:\Users\Brett\AppData\Local\Temp\ipykernel_26048\415629114.py:2: DataConversionWarni
ng: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples,), for example using ravel().
  forest_model.fit(df_num_workclass, df_salary_y)
```

Out[18]:

RandomForestClassifier

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

In [19]:
```
# Find predictions from tree model
workclass_decision_tree_predictions = forest_model.predict(golden_num_workclass)
```

In [20]:
```
# Precision, Recall, and F1,
print(classification_report(golden_salary_y, workclass_decision_tree_predictions))
# Confusion Matrix
print(confusion_matrix(golden_salary_y, workclass_decision_tree_predictions))
```

```
                precision    recall  f1-score   support

           0.0       0.81      0.99      0.89     12435
           1.0       0.93      0.26      0.40      3846

    accuracy                           0.82     16281
   macro avg       0.87      0.63      0.65     16281
weighted avg       0.84      0.82      0.78     16281

[[12362    73]
 [ 2860   986]]
```

In [21]:
```
# Here, we see that adding variables for the workclasses increased precision (.82 to .
```

In [22]:
```
# Encoding Education
## df
onehot_encoder = OneHotEncoder(handle_unknown='ignore')
onehot_encoder.fit(df[['education']])
df_new_edu = onehot_encoder.fit_transform(df[['education']]).toarray()
df_new_edu = pd.DataFrame(df_new_edu, columns = onehot_encoder.categories_[0])
df_new_edu

## golden
onehot_encoder = OneHotEncoder(handle_unknown='ignore')
onehot_encoder.fit(golden[['education']])
golden_new_edu = onehot_encoder.fit_transform(golden[['education']]).toarray()
golden_new_edu = pd.DataFrame(golden_new_edu, columns = onehot_encoder.categories_[0])
golden_new_edu
```

Out[22]:

	10th	11th	12th	1st-4th	5th-6th	7th-8th	9th	Assoc-acdm	Assoc-voc	Bachelors	Doctorate	HS-grad	Masters
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
16276	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
16277	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
16278	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
16279	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
16280	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

16281 rows × 16 columns

In [23]:
```
# Join to Original
# DF
df_num
df_num_edu = df_num_workclass.join(df_new_edu)
```

```
df_num_edu
# Golden
golden_num
golden_num_edu = golden_num_workclass.join(golden_new_edu)
golden_num_edu
```

Out[23]:

	age	education-num	capital-gain	capital-loss	hours-per-week	?	Federal-gov	Local-gov	Never-worked	Private	...	9th	As a
0	25	7	0	0	40	0.0	0.0	0.0	0.0	1.0	...	0.0	
1	38	9	0	0	50	0.0	0.0	0.0	0.0	1.0	...	0.0	
2	28	12	0	0	40	0.0	0.0	1.0	0.0	0.0	...	0.0	
3	44	10	7688	0	40	0.0	0.0	0.0	0.0	1.0	...	0.0	
4	18	10	0	0	30	1.0	0.0	0.0	0.0	0.0	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
16276	39	13	0	0	36	0.0	0.0	0.0	0.0	1.0	...	0.0	
16277	64	9	0	0	40	1.0	0.0	0.0	0.0	0.0	...	0.0	
16278	38	13	0	0	50	0.0	0.0	0.0	0.0	1.0	...	0.0	
16279	44	13	5455	0	40	0.0	0.0	0.0	0.0	1.0	...	0.0	
16280	35	13	0	0	60	0.0	0.0	0.0	0.0	0.0	...	0.0	

16281 rows × 30 columns

In [24]:
```
# Constructing tree model off of new DF with education
forest_model.fit(df_num_edu, df_salary_y)
```

```
C:\Users\Brett\AppData\Local\Temp\ipykernel_26048\1742727299.py:2: DataConversionWarn
ing: A column-vector y was passed when a 1d array was expected. Please change the sha
pe of y to (n_samples,), for example using ravel().
  forest_model.fit(df_num_edu, df_salary_y)
```

Out[24]:

RandomForestClassifier

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

In [25]:
```
# Find predictions from tree model
edu_decision_tree_predictions = forest_model.predict(golden_num_edu)
```

In [26]:
```
# Precision, Recall, and F1,
print(classification_report(golden_salary_y, edu_decision_tree_predictions))
# Confusion Matrix
print(confusion_matrix(golden_salary_y, edu_decision_tree_predictions))
```

```
                precision    recall  f1-score   support

           0.0       0.82      0.98      0.89     12435
           1.0       0.86      0.30      0.45      3846

    accuracy                           0.82     16281
   macro avg       0.84      0.64      0.67     16281
weighted avg       0.83      0.82      0.79     16281

[[12245   190]
 [ 2687  1159]]
```

In [27]:
```
# Here, we see the key statistics mostly unchanged, with a slight decrease in precisic
```

In [28]:
```
# Encoding Occuptation
## df
onehot_encoder = OneHotEncoder(handle_unknown='ignore')
onehot_encoder.fit(df[['occupation']])
df_new_occ = onehot_encoder.fit_transform(df[['occupation']]).toarray()
df_new_occ = pd.DataFrame(df_new_occ, columns = onehot_encoder.categories_[0])
df_new_occ

## golden
onehot_encoder = OneHotEncoder(handle_unknown='ignore')
onehot_encoder.fit(golden[['occupation']])
golden_new_occ = onehot_encoder.fit_transform(golden[['occupation']]).toarray()
golden_new_occ = pd.DataFrame(golden_new_occ, columns = onehot_encoder.categories_[0])
golden_new_occ
```

Out[28]:

	?	Adm-clerical	Armed-Forces	Craft-repair	Exec-managerial	Farming-fishing	Handlers-cleaners	Machine-op-inspct	Other-service	Priv-house-serv	sp
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
16276	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
16277	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
16278	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
16279	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
16280	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	

16281 rows × 15 columns

In [29]:
```
# Join to Original
# DF
df_num
```

```
df_num_occ = df_num_edu.join(df_num_edu, lsuffix='edu', rsuffix='occ')
df_num_occ
# Golden
golden_num
golden_num_occ = golden_num_edu.join(golden_num_edu,lsuffix='edu', rsuffix='occ')
golden_num_occ
```

Out[29]:

	ageedu	education-numedu	capital-gainedu	capital-lossedu	hours-per-weekedu	? edu	Federal-govedu	Local-govedu	Never-workededu	Privatee
0	25	7	0	0	40	0.0	0.0	0.0	0.0	
1	38	9	0	0	50	0.0	0.0	0.0	0.0	
2	28	12	0	0	40	0.0	0.0	1.0	0.0	
3	44	10	7688	0	40	0.0	0.0	0.0	0.0	
4	18	10	0	0	30	1.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
16276	39	13	0	0	36	0.0	0.0	0.0	0.0	
16277	64	9	0	0	40	1.0	0.0	0.0	0.0	
16278	38	13	0	0	50	0.0	0.0	0.0	0.0	
16279	44	13	5455	0	40	0.0	0.0	0.0	0.0	
16280	35	13	0	0	60	0.0	0.0	0.0	0.0	

16281 rows × 60 columns

In [30]:
```
# Constructing tree model off of new DF with education
forest_model.fit(df_num_occ, df_salary_y)
```

```
C:\Users\Brett\AppData\Local\Temp\ipykernel_26048\1067546612.py:2: DataConversionWarn
ing: A column-vector y was passed when a 1d array was expected. Please change the sha
pe of y to (n_samples,), for example using ravel().
  forest_model.fit(df_num_occ, df_salary_y)
```

Out[30]:

RandomForestClassifier

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

In [31]:
```
# Find predictions from tree model
occ_decision_tree_predictions = forest_model.predict(golden_num_occ)
```

In [32]:
```
# Precision, Recall, and F1,
print(classification_report(golden_salary_y, occ_decision_tree_predictions))
# Confusion Matrix
print(confusion_matrix(golden_salary_y, occ_decision_tree_predictions))
```

```
                   precision    recall  f1-score   support

              0.0       0.82      0.98      0.90     12435
              1.0       0.86      0.30      0.45      3846

         accuracy                           0.82     16281
        macro avg       0.84      0.64      0.67     16281
     weighted avg       0.83      0.82      0.79     16281

     [[12241   194]
      [ 2673  1173]]
```

In [33]:
```
# In this model, we see the statistics mostly unchanged; however from the confusion ma
#  true positives - possibly due to overfitting or that the new columns are not predic
```