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 [144…
```
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 [145…
```
df = pd.read_csv('../data/adult.data', index_col=False)
```

In [146…
```
golden = pd.read_csv('../data/adult.test', index_col=False)
```

In [147…
```
golden.head()
```

Out[147]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	cap
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	7
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	

In [148…
```
df.head()
```

Out[148]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	cap
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 [149…

```
df.columns
```

Out[149]:

```
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 [150…

```
from sklearn import preprocessing
```

In [151…

```
enc = preprocessing.OrdinalEncoder()
```

In [152…

```
transform_columns = ['sex']
non_num_columns = ['workclass', 'education', 'marital-status',
                   'occupation', 'relationship', 'race', 'sex',
                   'native-country']
```

In [153…

```
pd.get_dummies(df[transform_columns]).head()
```

Out[153]:

	sex_ Female	sex_ Male
0	0	1
1	0	1
2	0	1
3	0	1
4	1	0

In [154…

```
x = df.copy()

x = pd.concat([x.drop(non_num_columns, axis=1),
               pd.get_dummies(df[transform_columns])], axis=1,)

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

In [155…

```
x.head()
```

Out[155]:

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

In [156…

```
xt = golden.copy()

xt = pd.concat([xt.drop(non_num_columns, axis=1),
               pd.get_dummies(golden[transform_columns])], axis=1,)

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

In [157…
```
xt.salary.value_counts()
```
Out[157]:
```
0.0    12435
1.0     3846
Name: salary, dtype: int64
```
In [158…
```
enc.categories_
```
Out[158]:
```
[array([' <=50K.', ' >50K.'], dtype=object)]
```
In [159…
```
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 [160…
```
model = RandomForestClassifier(criterion='entropy')
```
In [161…
```
model = DecisionTreeClassifier(criterion='entropy', max_depth=None)
```
In [162…
```
model.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)
```
Out[162]:
```
DecisionTreeClassifier(criterion='entropy')
```
In [163…
```
model.tree_.node_count
```
Out[163]:
```
8325
```
In [164…
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```
Out[164]:
```
[('age', 0.3225890802977448),
 ('education-num', 0.1607231138360648),
 ('capital-gain', 0.2268185889866726),
 ('capital-loss', 0.0789379219010595),
 ('hours-per-week', 0.15539844295130806),
 ('sex_ Female', 0.05433887624806965),
 ('sex_ Male', 0.0011939757790807106)]
```
In [165…
```
list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))
```
Out[165]:
```
[('age', 0.3225890802977448),
 ('education-num', 0.1607231138360648),
 ('capital-gain', 0.2268185889866726),
 ('capital-loss', 0.0789379219010595),
 ('hours-per-week', 0.15539844295130806),
 ('sex_ Female', 0.05433887624806965),
 ('sex_ Male', 0.0011939757790807106)]
```
In [166…
```
x.drop(['fnlwgt','salary'], axis=1).head()
```
Out[166]:

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

In [167…
```
set(x.columns) - set(xt.columns)
```

Out[167]:
```
set()
```

In [168…
```
list(x.drop('salary', axis=1).columns)
```

Out[168]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'sex_ Female',
 'sex_ Male']
```

In [169…
```
predictions = model.predict(xt.drop(['fnlwgt','salary'], axis=1))
predictionsx = model.predict(x.drop(['fnlwgt','salary'], axis=1))
```

In [170…
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [171…
```
accuracy_score(xt.salary, predictions)
```

Out[171]:
```
0.8202813095018734
```

In [172…
```
accuracy_score(xt.salary, predictions)
```

Out[172]:
```
0.8202813095018734
```

In [173…
```
confusion_matrix(xt.salary, predictions)
```

Out[173]:
```
array([[11460,   975],
       [ 1951,  1895]])
```

In [174…
```
print(classification_report(xt.salary, predictions))
```

```
              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
```

In [175…
```
print(classification_report(xt.salary, predictions))
```

```
              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
```

In [176…
```
accuracy_score(x.salary, predictionsx)
```

Out[176]:
```
0.8955806025613464
```

```
In [177…  confusion_matrix(x.salary, predictionsx)

Out[177]: array([[24097,   623],
                 [ 2777,  5064]])
```

```
In [178…  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 [179…  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. Start with only numerical features/columns.

1. Show the RandomForest outperforms the DecisionTree for a fixed `max_depth` by training using the train set and `precision`, `recall`, `f1` on golden-test set.

2. For RandomForest or DecisionTree and using the `adult` dataset, systematically add new columns, one by one, that are non-numerical but converted using the feature-extraction techniques we learned. Show [`precision`, `recall`, `f1`] for each additional feature added.

3. Optional: Using gridSearch find the most optimal parameters for your model

Warning: this can be computationally intensive and may take some time.

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://scikit-learn.org/stable/modules/grid_search.html

In [180…
```
adult = df.copy().drop(['fnlwgt'], axis=1)
adult.head()
```

Out[180]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	cap
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	

In [181…
```
gold = golden.copy().drop(['fnlwgt'], axis=1)
gold.head()
```

Out[181]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	cap
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	
4	18	?	Some-college	10	Never-married	?	Own-child	White	Female	0	

In [182…
```
adult.dtypes
```

Out[182]:
```
age                int64
workclass         object
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
```

```
capital-gain        int64
capital-loss        int64
hours-per-week      int64
native-country     object
salary             object
dtype: object
```

1. Show the RandomForest outperforms the DecisionTree for a fixed max_depth by training using the train set and precision, recall, f1 on golden-test set.

In [183…

```
x1 = adult.copy()
x1 = pd.concat([x1.drop(non_num_columns, axis=1),
               pd.get_dummies(adult[transform_columns])], axis=1,)
x1["salary"] = enc.fit_transform(adult[["salary"]])
x1.head()
```

Out[183]:

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

In [184…

```
t1 = gold.copy()
t1 = pd.concat([t1.drop(non_num_columns, axis=1),
               pd.get_dummies(gold[transform_columns])], axis=1,)
t1["salary"] = enc.fit_transform(gold[["salary"]])
t1.head()
```

Out[184]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_ Female	sex_ Male
0	25	7	0	0	40	0.0	0	1
1	38	9	0	0	50	0.0	0	1
2	28	12	0	0	40	1.0	0	1
3	44	10	7688	0	40	1.0	0	1
4	18	10	0	0	30	0.0	1	0

In [185…

```
dt1 = DecisionTreeClassifier(criterion='entropy', max_depth=5)
rf1 = RandomForestClassifier(criterion='entropy', max_depth=5)
```

In [186…

```
dt1.fit(x1.drop(['salary'],axis=1),x1.salary)
rf1.fit(x1.drop(['salary'],axis=1),x1.salary)
```

Out[186]:

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

In [187…

```
list(zip(x1.drop(['salary'], axis=1).columns, dt1.feature_importances_))
```

Out[187]:

```
[('age', 0.27942432987238175),
 ('education-num', 0.21268599174904443),
 ('capital-gain', 0.3524074865400652),
 ('capital-loss', 0.04235939343563895),
 ('hours-per-week', 0.008875209903870702),
```

```
('sex_ Female', 0.0),
 ('sex_ Male', 0.10424758849899901)]
```

In [188…
```
list(zip(x1.drop(['salary'], axis=1).columns, rf1.feature_importances_))
```

Out[188]:
```
[('age', 0.25060207498456627),
 ('education-num', 0.2100045738552291),
 ('capital-gain', 0.2834041907641028),
 ('capital-loss', 0.05455832340210596),
 ('hours-per-week', 0.0785636245268627),
 ('sex_ Female', 0.06803882879627765),
 ('sex_ Male', 0.05482838367085557)]
```

In [189…
```
dtpred1 = dt1.predict(t1.drop(['salary'],axis=1))
rfpred1 = rf1.predict(t1.drop(['salary'],axis=1))
```

In [190…
```
accuracy_score(t1.salary, dtpred1)
```

Out[190]:
```
0.8200356243473989
```

In [191…
```
accuracy_score(t1.salary, rfpred1)
```

Out[191]:
```
0.8341010994410663
```

In [192…
```
confusion_matrix(t1.salary, dtpred1)
```

Out[192]:
```
array([[11457,   978],
       [ 1952,  1894]])
```

In [193…
```
confusion_matrix(t1.salary, rfpred1)
```

Out[193]:
```
array([[12070,   365],
       [ 2336,  1510]])
```

In [194…
```
print(classification_report(t1.salary, dtpred1))
```

```
              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
```

In [195…
```
print(classification_report(t1.salary, rfpred1))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.97      0.90     12435
         1.0       0.81      0.39      0.53      3846

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

2. For RandomForest or DecisionTree and using the adult dataset, systematically add new columns, one by one, that are non-numerical but converted using the feature-extraction

techniques we learned. Show [precision, recall, f1] for each additional feature added.

In [196…

```
adult.head()
```

Out[196]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	ca
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	

In [197…

```
adult.workclass.value_counts()
```

Out[197]:

```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

In [198…

```
x2 = x1.copy()
x2['workclass'] = enc.fit_transform(adult[['workclass']])
x2.head()
```

Out[198]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass
0	39	13	2174	0	40	0.0	0	1	7.0
1	50	13	0	0	13	0.0	0	1	6.0
2	38	9	0	0	40	0.0	0	1	4.0
3	53	7	0	0	40	0.0	0	1	4.0
4	28	13	0	0	40	0.0	1	0	4.0

In [199…

```
t2 = t1.copy()
t2['workclass'] = enc.fit_transform(gold[['workclass']])
t2.head()
```

Out[199]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass
0	25	7	0	0	40	0.0	0	1	4.0

1	38	9	0	0	50	0.0	0	1	4.0
2	28	12	0	0	40	1.0	0	1	2.0
3	44	10	7688	0	40	1.0	0	1	4.0
4	18	10	0	0	30	0.0	1	0	0.0

In [200…

```
rf2 = rf1.fit(x2.drop(['salary'],axis=1),x2.salary)
rfpred2 = rf2.predict(t2.drop(['salary'],axis=1))
print(classification_report(t2.salary, rfpred2))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.98      0.90     12435
         1.0       0.82      0.38      0.52      3846

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

In [201…

```
adult.education.value_counts()
```

Out[201]:

```
 HS-grad         10501
 Some-college     7291
 Bachelors        5355
 Masters          1723
 Assoc-voc        1382
 11th             1175
 Assoc-acdm       1067
 10th              933
 7th-8th           646
 Prof-school       576
 9th               514
 12th              433
 Doctorate         413
 5th-6th           333
 1st-4th           168
 Preschool          51
Name: education, dtype: int64
```

In [202…

```
x3 = x2.copy()
x3['education'] = enc.fit_transform(adult[['education']])
x3.head()
```

Out[202]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass	education
0	39	13	2174	0	40	0.0	0	1	7.0	9.0
1	50	13	0	0	13	0.0	0	1	6.0	9.0
2	38	9	0	0	40	0.0	0	1	4.0	11.0
3	53	7	0	0	40	0.0	0	1	4.0	1.0
4	28	13	0	0	40	0.0	1	0	4.0	9.0

In [203…

```
t3 = t2.copy()
t3['education'] = enc.fit_transform(gold[['education']])
t3.head()
```

Out[203]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass	education
0	25	7	0	0	40	0.0	0	1	4.0	1.0

1	38	9	0	0	50	0.0	0	1	4.0	11.0
2	28	12	0	0	40	1.0	0	1	2.0	7.0
3	44	10	7688	0	40	1.0	0	1	4.0	15.0
4	18	10	0	0	30	0.0	1	0	0.0	15.0

In [204…

```
rf3 = rf1.fit(x3.drop(['salary'],axis=1),x3.salary)
rfpred3 = rf3.predict(t3.drop(['salary'],axis=1))
print(classification_report(t3.salary, rfpred3))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.97      0.90     12435
         1.0       0.78      0.39      0.52      3846

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

In [205…

```
adult['marital-status'].value_counts()
```

Out[205]:

```
Married-civ-spouse       14976
Never-married            10683
Divorced                  4443
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: marital-status, dtype: int64
```

In [206…

```
x4 = x3.copy()
x4['marital-status'] = enc.fit_transform(adult[['marital-status']])
x4.head()
```

Out[206]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass	education	marital-status
0	39	13	2174	0	40	0.0	0	1	7.0	9.0	4.0
1	50	13	0	0	13	0.0	0	1	6.0	9.0	2.0
2	38	9	0	0	40	0.0	0	1	4.0	11.0	0.0
3	53	7	0	0	40	0.0	0	1	4.0	1.0	2.0
4	28	13	0	0	40	0.0	1	0	4.0	9.0	2.0

In [207…

```
t4 = t3.copy()
t4['marital-status'] = enc.fit_transform(gold[['marital-status']])
t4.head()
```

Out[207]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass	education	marital-status
0	25	7	0	0	40	0.0	0	1	4.0	1.0	4.0
1	38	9	0	0	50	0.0	0	1	4.0	11.0	2.0
2	28	12	0	0	40	1.0	0	1	2.0	7.0	2.0
3	44	10	7688	0	40	1.0	0	1	4.0	15.0	2.0

4	18	10	0	0	30	0.0	1	0	0.0	15.0	4.0

In [208…
```
rf4 = rf1.fit(x4.drop(['salary'],axis=1),x4.salary)
rfpred4 = rf4.predict(t4.drop(['salary'],axis=1))
print(classification_report(t4.salary, rfpred4))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.96      0.91     12435
         1.0       0.80      0.48      0.60      3846

    accuracy                           0.85     16281
   macro avg       0.83      0.72      0.75     16281
weighted avg       0.84      0.85      0.83     16281
```

In [209…
```
adult.occupation.value_counts()
```

Out[209]:
```
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
 ?                    1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64
```

In [210…
```
x5 = x4.copy()
x5['occupation'] = enc.fit_transform(adult[['occupation']])
x5.head()
```

Out[210]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_ Female	sex_ Male	workclass	education	marital-status	occ
0	39	13	2174	0	40	0.0	0	1	7.0	9.0	4.0	
1	50	13	0	0	13	0.0	0	1	6.0	9.0	2.0	
2	38	9	0	0	40	0.0	0	1	4.0	11.0	0.0	
3	53	7	0	0	40	0.0	0	1	4.0	1.0	2.0	
4	28	13	0	0	40	0.0	1	0	4.0	9.0	2.0	

In [211…
```
t5 = t4.copy()
t5['occupation'] = enc.fit_transform(gold[['occupation']])
t5.head()
```

Out[211]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_ Female	sex_ Male	workclass	education	marital-status	occ
0	25	7	0	0	40	0.0	0	1	4.0	1.0	4.0	
1	38	9	0	0	50	0.0	0	1	4.0	11.0	2.0	
2	28	12	0	0	40	1.0	0	1	2.0	7.0	2.0	

3	44	10	7688	0	40	1.0	0	1	4.0	15.0	2.0	
4	18	10	0	0	30	0.0	1	0	0.0	15.0	4.0	

In [212…

```
rf5 = rf1.fit(x5.drop(['salary'],axis=1),x5.salary)
rfpred5 = rf5.predict(t5.drop(['salary'],axis=1))
print(classification_report(t5.salary, rfpred5))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.96      0.91     12435
         1.0       0.78      0.49      0.60      3846

    accuracy                           0.85     16281
   macro avg       0.82      0.72      0.75     16281
weighted avg       0.84      0.85      0.83     16281
```

In [213…

```
adult.relationship.value_counts()
```

Out[213]:

```
 Husband           13193
 Not-in-family      8305
 Own-child          5068
 Unmarried          3446
 Wife               1568
 Other-relative      981
Name: relationship, dtype: int64
```

In [214…

```
x6 = x5.copy()
x6['relationship'] = enc.fit_transform(adult[['relationship']])
x6.head()
```

Out[214]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass	education	marital-status	occ
0	39	13	2174	0	40	0.0	0	1	7.0	9.0	4.0	
1	50	13	0	0	13	0.0	0	1	6.0	9.0	2.0	
2	38	9	0	0	40	0.0	0	1	4.0	11.0	0.0	
3	53	7	0	0	40	0.0	0	1	4.0	1.0	2.0	
4	28	13	0	0	40	0.0	1	0	4.0	9.0	2.0	

In [215…

```
t6 = t5.copy()
t6['relationship'] = enc.fit_transform(gold[['relationship']])
t6.head()
```

Out[215]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_Female	sex_Male	workclass	education	marital-status	occ
0	25	7	0	0	40	0.0	0	1	4.0	1.0	4.0	
1	38	9	0	0	50	0.0	0	1	4.0	11.0	2.0	
2	28	12	0	0	40	1.0	0	1	2.0	7.0	2.0	
3	44	10	7688	0	40	1.0	0	1	4.0	15.0	2.0	
4	18	10	0	0	30	0.0	1	0	0.0	15.0	4.0	

In [216…

```
rf6 = rf1.fit(x6.drop(['salary'],axis=1),x6.salary)
rfpred6 = rf6.predict(t6.drop(['salary'],axis=1))
```

```
print(classification_report(t6.salary, rfpred6))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.96      0.91     12435
         1.0       0.80      0.49      0.61      3846

    accuracy                           0.85     16281
   macro avg       0.83      0.73      0.76     16281
weighted avg       0.84      0.85      0.84     16281
```

In [217…

```
adult.race.value_counts()
```

Out[217]:

```
 White                 27816
 Black                  3124
 Asian-Pac-Islander     1039
 Amer-Indian-Eskimo      311
 Other                   271
Name: race, dtype: int64
```

In [218…

```
x7 = x6.copy()
x7 = pd.concat([x7,
               pd.get_dummies(adult['race'])], axis=1)
x7.head()
```

Out[218]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_ Female	sex_ Male	workclass	education	marital-status	occ
0	39	13	2174	0	40	0.0	0	1	7.0	9.0	4.0	
1	50	13	0	0	13	0.0	0	1	6.0	9.0	2.0	
2	38	9	0	0	40	0.0	0	1	4.0	11.0	0.0	
3	53	7	0	0	40	0.0	0	1	4.0	1.0	2.0	
4	28	13	0	0	40	0.0	1	0	4.0	9.0	2.0	

In [219…

```
t7 = t6.copy()
t7 = pd.concat([t7,
               pd.get_dummies(gold['race'])], axis=1)
t7.head()
```

Out[219]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_ Female	sex_ Male	workclass	education	marital-status	occ
0	25	7	0	0	40	0.0	0	1	4.0	1.0	4.0	
1	38	9	0	0	50	0.0	0	1	4.0	11.0	2.0	
2	28	12	0	0	40	1.0	0	1	2.0	7.0	2.0	
3	44	10	7688	0	40	1.0	0	1	4.0	15.0	2.0	
4	18	10	0	0	30	0.0	1	0	0.0	15.0	4.0	

In [220…

```
rf7 = rf1.fit(x7.drop(['salary'],axis=1),x7.salary)
rfpred7 = rf7.predict(t7.drop(['salary'],axis=1))
print(classification_report(t7.salary, rfpred7))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.96      0.91     12435
         1.0       0.79      0.49      0.61      3846
```

```
    accuracy                           0.85     16281
   macro avg       0.82      0.73      0.76     16281
weighted avg       0.84      0.85      0.84     16281
```

In [221…

```
list(zip(x7.drop(['salary'], axis=1).columns, rf7.feature_importances_))
```

Out[221]:

```
[('age', 0.06950563975752137),
 ('education-num', 0.13984502540377203),
 ('capital-gain', 0.2025029163128069),
 ('capital-loss', 0.025746355853208815),
 ('hours-per-week', 0.04426595319204749),
 ('sex_ Female', 0.020204875460454814),
 ('sex_ Male', 0.010260486788901894),
 ('workclass', 0.001467851033114902),
 ('education', 0.026506516228131535),
 ('marital-status', 0.1920228653408138),
 ('occupation', 0.010984498542867831),
 ('relationship', 0.25581504899509583),
 (' Amer-Indian-Eskimo', 0.00016098402626058223),
 (' Asian-Pac-Islander', 0.0001028875473341934),
 (' Black', 0.0002810218158809956),
 (' Other', 9.996609648700666e-05),
 (' White', 0.0002271076053000775)]
```

In [222…

```
adult['native-country'].value_counts()
```

Out[222]:

```
United-States                 29170
Mexico                          643
?                               583
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        64
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
France                           29
Greece                           29
Ecuador                          28
Ireland                          24
Hong                             20
Cambodia                         19
Trinadad&Tobago                  19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
```

```
 Honduras                          13
 Hungary                           13
 Scotland                          12
 Holand-Netherlands                 1
Name: native-country, dtype: int64
```

In [223…

```
x8 = x7.copy()
x8['native-country'] = enc.fit_transform(adult[['native-country']])
x8.head()
```

Out[223]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_ Female	sex_ Male	workclass	education	marital-status	occ
0	39	13	2174	0	40	0.0	0	1	7.0	9.0	4.0	
1	50	13	0	0	13	0.0	0	1	6.0	9.0	2.0	
2	38	9	0	0	40	0.0	0	1	4.0	11.0	0.0	
3	53	7	0	0	40	0.0	0	1	4.0	1.0	2.0	
4	28	13	0	0	40	0.0	1	0	4.0	9.0	2.0	

In [224…

```
t8 = t7.copy()
t8['native-country'] = enc.fit_transform(gold[['native-country']])
t8.head()
```

Out[224]:

	age	education-num	capital-gain	capital-loss	hours-per-week	salary	sex_ Female	sex_ Male	workclass	education	marital-status	occ
0	25	7	0	0	40	0.0	0	1	4.0	1.0	4.0	
1	38	9	0	0	50	0.0	0	1	4.0	11.0	2.0	
2	28	12	0	0	40	1.0	0	1	2.0	7.0	2.0	
3	44	10	7688	0	40	1.0	0	1	4.0	15.0	2.0	
4	18	10	0	0	30	0.0	1	0	0.0	15.0	4.0	

In [225…

```
rf8 = rf1.fit(x8.drop(['salary'],axis=1),x8.salary)
rfpred8 = rf8.predict(t8.drop(['salary'],axis=1))
print(classification_report(t8.salary, rfpred8))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.96      0.91     12435
         1.0       0.79      0.48      0.60      3846

    accuracy                           0.85     16281
   macro avg       0.82      0.72      0.75     16281
weighted avg       0.84      0.85      0.83     16281
```

In [226…

```
list(zip(x8.drop(['salary'], axis=1).columns, rf8.feature_importances_))
```

Out[226]:

```
[('age', 0.09315313503748186),
 ('education-num', 0.141954314016355),
 ('capital-gain', 0.2013202078235829),
 ('capital-loss', 0.023589564343195653),
 ('hours-per-week', 0.04218309497075345),
 ('sex_ Female', 0.024070322070793848),
 ('sex_ Male', 0.02389142046735729 5),
 ('workclass', 0.002555344425558234),
 ('education', 0.025164904954484524),
```

```
 ('marital-status', 0.17691583151348625),
 ('occupation', 0.010923233799191856),
 ('relationship', 0.23201689419975413),
 (' Amer-Indian-Eskimo', 6.575017624484862e-05),
 (' Asian-Pac-Islander', 7.217424952946028e-05),
 (' Black', 0.0007063204186603916),
 (' Other', 6.620254617335375e-05),
 (' White', 0.000435496119348729),
 ('native-country', 0.0009157888680481677)]
```

3. Optional: Using gridSearch find the most optimal parameters for your model

Warning: this can be computationally intensive and may take some time.

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://scikit-learn.org/stable/modules/grid_search.html

In [227…

```
rf.get_params().keys()
```

Out[227]:

```
dict_keys(['bootstrap', 'ccp_alpha', 'class_weight', 'criterion', 'max_depth', 'max_features', 'max_leaf_nodes', 'max_samples', 'min_impurity_decrease', 'min_samples_leaf', 'min_samples_split', 'min_weight_fraction_leaf', 'n_estimators', 'n_jobs', 'oob_score', 'random_state', 'verbose', 'warm_start'])
```

In [228…

```
from sklearn.model_selection import GridSearchCV
rf = RandomForestClassifier(criterion='entropy')
param_grid_rfc = [{
    'max_depth':[2, 3, 4, 5, 6, 7, 8],
    'max_features':[2, 3, 4, 5, 6, 7, 8]
}]
clf = GridSearchCV(estimator=rf,
                     param_grid = param_grid_rfc,
                     scoring='accuracy',
                     cv=10,
                     refit=True,
                     n_jobs=1)
clf.fit(x8.drop(['salary'],axis=1),x8.salary)
print(clf.best_score_)
print(clf.best_params_)
clfRFC = clf.best_estimator_
print('Test accuracy: %.3f' % clfRFC.score(t8.drop(['salary'],axis=1), t8.salary))
```

```
0.8560551116891307
{'max_depth': 8, 'max_features': 8}
Test accuracy: 0.857
```