Ben Roth Week 6 - Jupyter Notebook (Trees Classification)

March 6, 2020

- 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]:
```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

In [2]:
```
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 [3]:
```
df = pd.read_csv('../data/adult.data', index_col=False)
```

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

In [5]:
```
golden.head()
```

Out[5]:
```
   age  workclass  fnlwgt     education  education-num      marital-status  \
0   25    Private  226802          11th              7       Never-married
1   38    Private   89814       HS-grad              9  Married-civ-spouse
2   28  Local-gov  336951    Assoc-acdm             12  Married-civ-spouse
3   44    Private  160323  Some-college             10  Married-civ-spouse
4   18          ?  103497  Some-college             10       Never-married

          occupation relationship   race     sex  capital-gain  \
0  Machine-op-inspct    Own-child  Black    Male             0
1    Farming-fishing      Husband  White    Male             0
2    Protective-serv      Husband  White    Male             0
3  Machine-op-inspct      Husband  Black    Male          7688
4                  ?    Own-child  White  Female             0

   capital-loss  hours-per-week  native-country  salary
0             0              40   United-States  <=50K.
1             0              50   United-States  <=50K.
2             0              40   United-States   >50K.
3             0              40   United-States   >50K.
4             0              30   United-States  <=50K.
```

In [6]: df.head()

Out[6]:
```
   age          workclass  fnlwgt  education  education-num  \
0   39          State-gov   77516  Bachelors             13
1   50   Self-emp-not-inc   83311  Bachelors             13
2   38            Private  215646    HS-grad              9
3   53            Private  234721       11th              7
4   28            Private  338409  Bachelors             13

       marital-status          occupation   relationship   race      sex  \
0       Never-married        Adm-clerical  Not-in-family  White     Male
1  Married-civ-spouse     Exec-managerial        Husband  White     Male
2            Divorced   Handlers-cleaners  Not-in-family  White     Male
3  Married-civ-spouse   Handlers-cleaners        Husband  Black     Male
4  Married-civ-spouse      Prof-specialty           Wife  Black   Female

   capital-gain  capital-loss  hours-per-week  native-country  salary
0          2174             0              40   United-States   <=50K
1             0             0              13   United-States   <=50K
2             0             0              40   United-States   <=50K
3             0             0              40   United-States   <=50K
4             0             0              40            Cuba   <=50K
```

In [7]: df.columns

Out[7]:
```
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 [8]:
```python
from sklearn import preprocessing
```

In [9]:
```python
enc = preprocessing.OrdinalEncoder()
```

In [10]:
```python
transform_columns = ['sex']
non_num_columns = ['workclass', 'education', 'marital-status',
                   'occupation', 'relationship', 'race', 'sex',
                   'native-country']
```

In [11]:
```python
pd.get_dummies(df[transform_columns]).head()
```

Out[11]:
```
   sex_ Female  sex_ Male
0            0          1
1            0          1
2            0          1
3            0          1
4            1          0
```

In [12]:
```python
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 [13]:
```python
x.head()
```

Out[13]:
```
   age  fnlwgt  education-num  capital-gain  capital-loss  hours-per-week  \
0   39   77516             13          2174             0              40
1   50   83311             13             0             0              13
2   38  215646              9             0             0              40
3   53  234721              7             0             0              40
4   28  338409             13             0             0              40

   salary  sex_ Female  sex_ Male
0     0.0            0          1
1     0.0            0          1
2     0.0            0          1
3     0.0            0          1
4     0.0            1          0
```

In [14]:
```python
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 [15]:
```python
xt.salary.value_counts()
```

Out[15]:
```
0.0    12435
1.0     3846
Name: salary, dtype: int64
```

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

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

In [17]:
```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 [18]:
```python
model = RandomForestClassifier(criterion='entropy')
```

In [19]:
```python
model = DecisionTreeClassifier(criterion='entropy', max_depth=None)
```

```
In [20]: model.fit(x.drop(['fnlwgt','salary'], axis=1), x.salary)

Out[20]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort=False,
                      random_state=None, splitter='best')

In [21]: model.tree_.node_count

Out[21]: 8321

In [22]: list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))

Out[22]: [('age', 0.32291554471892336),
          ('education-num', 0.16238431110494822),
          ('capital-gain', 0.22779935087425895),
          ('capital-loss', 0.07844540364173368),
          ('hours-per-week', 0.15302090309108082),
          ('sex_ Female', 0.054375187001441436),
          ('sex_ Male', 0.0010592995676135919)]

In [23]: list(zip(x.drop(['fnlwgt','salary'], axis=1).columns, model.feature_importances_))

Out[23]: [('age', 0.32291554471892336),
          ('education-num', 0.16238431110494822),
          ('capital-gain', 0.22779935087425895),
          ('capital-loss', 0.07844540364173368),
          ('hours-per-week', 0.15302090309108082),
          ('sex_ Female', 0.054375187001441436),
          ('sex_ Male', 0.0010592995676135919)]

In [24]: x.drop(['fnlwgt','salary'], axis=1).head()

Out[24]:    age  education-num  capital-gain  capital-loss  hours-per-week  \
         0   39             13          2174             0              40
         1   50             13             0             0              13
         2   38              9             0             0              40
         3   53              7             0             0              40
         4   28             13             0             0              40

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

In [25]: set(x.columns) - set(xt.columns)
```

Out[25]: set()

In [26]:
```
list(x.drop('salary', axis=1).columns)
```

Out[26]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'sex_ Female',
 'sex_ Male']
```

In [27]:
```
predictions = model.predict(xt.drop(['fnlwgt','salary'], axis=1))
predictionsx = model.predict(x.drop(['fnlwgt','salary'], axis=1))
```

In [28]:
```
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix, auc, roc_curve
)
```

In [29]:
```
accuracy_score(xt.salary, predictions)
```

Out[29]: 0.8205269946563479

In [30]:
```
accuracy_score(xt.salary, predictions)
```

Out[30]: 0.8205269946563479

In [31]:
```
confusion_matrix(xt.salary, predictions)
```

Out[31]:
```
array([[11459,   976],
       [ 1946,  1900]], dtype=int64)
```

In [32]:
```
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      0.57      3846

    accuracy                           0.82     16281
   macro avg       0.76      0.71      0.73     16281
weighted avg       0.81      0.82      0.81     16281
```

In [33]:
```
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      0.57      3846

    accuracy                           0.82     16281
   macro avg       0.76      0.71      0.73     16281
weighted avg       0.81      0.82      0.81     16281
```

In [34]:
```
accuracy_score(x.salary, predictionsx)
```

Out[34]:
```
0.8955806025613464
```

In [35]:
```
confusion_matrix(x.salary, predictionsx)
```

Out[35]:
```
array([[24097,   623],
       [ 2777,  5064]], dtype=int64)
```

In [36]:
```
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 [37]:
```
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
```

1 For the following use the above `adult` dataset. Start with only numerical features/columns.

2 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 [38]: rf = RandomForestClassifier(criterion='entropy', max_depth = 10)
         dt =  DecisionTreeClassifier(criterion='entropy', max_depth = 10)

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

         df["salary"] = enc.fit_transform(df[["salary"]])
         golden["salary"] = enc.fit_transform(golden[["salary"]])

         df_drop = df.drop(columns = non_num_columns)

         gold_drop = golden.drop(columns = non_num_columns)

         rf.fit(df_drop.drop(['fnlwgt','salary'], axis=1), df_drop.salary)
         dt.fit(df_drop.drop(['fnlwgt','salary'], axis=1), df_drop.salary)

         preds_rf = rf.predict(gold_drop.drop(['fnlwgt','salary'], axis=1))
         preds_dt = dt.predict(gold_drop.drop(['fnlwgt','salary'], axis=1))
```

```
C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
In [39]: print('The Classification Report for the Random Forest Model is:')
         print(classification_report(gold_drop.salary, preds_rf))

         print('\n\nThe Classification Report for the Decision Model is:')
         print(classification_report(gold_drop.salary, preds_dt))
```

```
The Classification Report for the Random Forest Model is:
              precision    recall  f1-score   support

         0.0       0.85      0.96      0.90     12435
         1.0       0.78      0.44      0.56      3846

    accuracy                           0.84     16281
   macro avg       0.81      0.70      0.73     16281
weighted avg       0.83      0.84      0.82     16281
```

```
The Classification Report for the Decision Model is:
              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       0.78      0.71      0.74     16281
weighted avg       0.82      0.83      0.82     16281
```

3 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 [40]:
```
from sklearn.preprocessing import LabelEncoder
import numpy as np

df = pd.read_csv('../data/adult.data', index_col=False)
golden = pd.read_csv('../data/adult.test', index_col=False)

df["salary"] = enc.fit_transform(df[["salary"]])
golden["salary"] = enc.fit_transform(golden[["salary"]])

for col in non_num_columns:
    df[col] = LabelEncoder().fit_transform(df[col].values)
    golden[col] = LabelEncoder().fit_transform(golden[col].values)

    df_num = df.select_dtypes([np.number])
    gold_num = golden.select_dtypes([np.number])

    rf = RandomForestClassifier(criterion='entropy', max_depth = 10)
    dt =  DecisionTreeClassifier(criterion='entropy', max_depth = 10)

    rf.fit(df_num.drop(['fnlwgt','salary'], axis=1), df_num.salary)
    dt.fit(df_num.drop(['fnlwgt','salary'], axis=1), df_num.salary)

    preds_rf = rf.predict(gold_num.drop(['fnlwgt','salary'], axis=1))
    preds_dt = dt.predict(gold_num.drop(['fnlwgt','salary'], axis=1))

    print('The columns in the model data is: ', df_num.columns)
```

```
            print('The Classification Report for the Random Forest Model (adding ', col, ' to
            print(classification_report(gold_num.salary, preds_rf))

            print('\n\nThe Classification Report for the Decision Model (adding ', col, ' to
            print(classification_report(gold_num.salary, preds_dt))
```

```
C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education-num', 'capi
       'capital-loss', 'hours-per-week', 'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  workclass  to the data) is:
              precision    recall  f1-score   support

         0.0       0.85      0.96      0.90     12435
         1.0       0.79      0.43      0.55      3846

    accuracy                           0.84     16281
   macro avg       0.82      0.70      0.73     16281
weighted avg       0.83      0.84      0.82     16281




The Classification Report for the Decision Model (adding  workclass  to the data) is:
              precision    recall  f1-score   support

         0.0       0.85      0.95      0.90     12435
         1.0       0.74      0.47      0.57      3846

    accuracy                           0.84     16281
   macro avg       0.80      0.71      0.74     16281
weighted avg       0.83      0.84      0.82     16281



C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education', 'educatio
       'capital-gain', 'capital-loss', 'hours-per-week', 'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  education  to the data) is:
              precision    recall  f1-score   support

         0.0       0.85      0.96      0.90     12435
         1.0       0.77      0.45      0.56      3846
```

```
    accuracy                           0.84     16281
   macro avg       0.81      0.70      0.73     16281
weighted avg       0.83      0.84      0.82     16281
```

```
The Classification Report for the Decision Model (adding  education  to the data) is:
              precision    recall  f1-score   support

         0.0       0.85      0.95      0.90     12435
         1.0       0.75      0.47      0.57      3846

    accuracy                           0.84     16281
   macro avg       0.80      0.71      0.74     16281
weighted avg       0.83      0.84      0.82     16281
```

```
C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education', 'educatio
       'marital-status', 'capital-gain', 'capital-loss', 'hours-per-week',
       'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  marital-status  to the data) is
              precision    recall  f1-score   support

         0.0       0.87      0.96      0.91     12435
         1.0       0.79      0.53      0.63      3846

    accuracy                           0.86     16281
   macro avg       0.83      0.74      0.77     16281
weighted avg       0.85      0.86      0.84     16281
```

```
The Classification Report for the Decision Model (adding  marital-status  to the data) is:
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.91     12435
         1.0       0.71      0.64      0.68      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.78      0.79     16281
weighted avg       0.85      0.85      0.85     16281
```

```
C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education', 'education
       'marital-status', 'occupation', 'capital-gain', 'capital-loss',
       'hours-per-week', 'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  occupation  to the data) is:
              precision    recall  f1-score   support

         0.0       0.87      0.96      0.91     12435
         1.0       0.79      0.53      0.63      3846

    accuracy                           0.86     16281
   macro avg       0.83      0.74      0.77     16281
weighted avg       0.85      0.86      0.85     16281



The Classification Report for the Decision Model (adding  occupation  to the data) is:
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.91     12435
         1.0       0.71      0.65      0.68      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.78      0.79     16281
weighted avg       0.85      0.85      0.85     16281



C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education', 'education
       'marital-status', 'occupation', 'relationship', 'capital-gain',
       'capital-loss', 'hours-per-week', 'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  relationship  to the data) is:
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     12435
         1.0       0.79      0.54      0.64      3846
```

```
    accuracy                           0.86     16281
   macro avg       0.83      0.75      0.78     16281
weighted avg       0.85      0.86      0.85     16281
```

```
The Classification Report for the Decision Model (adding  relationship  to the data) is:
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.91     12435
         1.0       0.72      0.64      0.68      3846

    accuracy                           0.86     16281
   macro avg       0.80      0.78      0.79     16281
weighted avg       0.85      0.86      0.85     16281
```

```
C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education', 'education
       'marital-status', 'occupation', 'relationship', 'race', 'capital-gain',
       'capital-loss', 'hours-per-week', 'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  race  to the data) is:
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     12435
         1.0       0.79      0.54      0.64      3846

    accuracy                           0.86     16281
   macro avg       0.83      0.75      0.78     16281
weighted avg       0.85      0.86      0.85     16281
```

```
The Classification Report for the Decision Model (adding  race  to the data) is:
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.91     12435
         1.0       0.72      0.64      0.68      3846

    accuracy                           0.86     16281
   macro avg       0.80      0.78      0.79     16281
weighted avg       0.85      0.86      0.85     16281
```

```
C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education', 'education
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  sex  to the data) is:
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     12435
         1.0       0.79      0.56      0.65      3846

    accuracy                           0.86     16281
   macro avg       0.83      0.76      0.78     16281
weighted avg       0.85      0.86      0.85     16281



The Classification Report for the Decision Model (adding  sex  to the data) is:
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.91     12435
         1.0       0.72      0.64      0.68      3846

    accuracy                           0.86     16281
   macro avg       0.80      0.78      0.79     16281
weighted avg       0.85      0.86      0.85     16281



C:\Users\rothb13\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


The columns in the model data is:  Index(['age', 'workclass', 'fnlwgt', 'education', 'education
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'salary'],
      dtype='object')
The Classification Report for the Random Forest Model (adding  native-country  to the data) is
              precision    recall  f1-score   support

         0.0       0.87      0.96      0.91     12435
         1.0       0.79      0.53      0.63      3846

    accuracy                           0.86     16281
```

```
   macro avg       0.83      0.74      0.77     16281
weighted avg       0.85      0.86      0.85     16281
```

```
The Classification Report for the Decision Model (adding  native-country  to the data) is:
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.91     12435
         1.0       0.71      0.64      0.68      3846

    accuracy                           0.85     16281
   macro avg       0.80      0.78      0.79     16281
weighted avg       0.85      0.85      0.85     16281
```

4 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 [41]: from sklearn.model_selection import GridSearchCV
         param_grid = {'max_depth':np.arange(1, 26, 5),
                       'max_features':['auto','log2'],
                       'min_samples_leaf': np.arange(1, 10, 1)
                       }

         rfmodel = RandomForestClassifier(criterion='entropy', n_estimators = 50)

         rf_gridsearch = GridSearchCV(estimator=rfmodel, param_grid=param_grid, cv= 10)
         rf_gridsearch.fit(df_num.drop(['fnlwgt','salary'], axis=1), df_num.salary)

         preds_rfgrid = rf_gridsearch.predict(gold_num.drop(['fnlwgt','salary'], axis=1))

         print('The Classification Report for the Random Forest Model is:')
         print(classification_report(gold_num.salary, preds_rfgrid))
```

```
The Classification Report for the Random Forest Model is:
              precision    recall  f1-score   support

         0.0       0.88      0.94      0.91     12435
         1.0       0.77      0.60      0.67      3846

    accuracy                           0.86     16281
```

```
   macro avg       0.83      0.77      0.79     16281
weighted avg       0.86      0.86      0.86     16281
```