In [1]:
```
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
%matplotlib inline
import seaborn as sns
from sklearn.metrics import classification_report
```

In [2]:
```
train_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adul
test_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult
```

In [3]:
```
column_labels=['age','workingclass','fnlwgt','education', 'education_num', 'marital_st
```

In [4]:
```
train_set.columns=column_labels
```

In [5]:
```
#show the missing rows of ' ?' as the unknown values. going to drop them.
train_set.replace(' ?', np.nan).dropna().shape
```

Out[5]:
```
(15060, 15)
```

In [6]:
```
#perform the drop and save as new dataframes
train_nomiss=train_set.replace(' ?', np.nan).dropna()
```

In [7]:
```
#wage class column is dirty, clean with replace. getting rid of period. test to see it
test_nomiss.wage_class=test_nomiss.wage_class.replace({' <=50K.': ' <=50K', ' >50K.':'
test_nomiss.wage_class.unique()
```

Out[7]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [8]:
```
#XGBoosting
#Step 1: ordinal encoding to categoricals

combined_set=pd.concat([train_nomiss,test_nomiss], axis=0)

combined_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45222 entries, 0 to 16280
Data columns (total 15 columns):
age               45222 non-null int64
workingclass      45222 non-null object
fnlwgt            45222 non-null int64
education         45222 non-null object
education_num     45222 non-null int64
marital_status    45222 non-null object
occupation        45222 non-null object
relationship      45222 non-null object
race              45222 non-null object
sex               45222 non-null object
capital_gain      45222 non-null int64
capital_loss      45222 non-null int64
hours_per_week    45222 non-null int64
native_country    45222 non-null object
wage_class        45222 non-null object
dtypes: int64(6), object(9)
memory usage: 5.5+ MB
```

In [9]:
```
#use categorical codes from pandas.

for feat in combined_set.columns:
  if combined_set[feat].dtype =='object':
    combined_set[feat]=pd.Categorical(combined_set[feat]).codes #replaces str with int

combined_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45222 entries, 0 to 16280
Data columns (total 15 columns):
age               45222 non-null int64
workingclass      45222 non-null int8
fnlwgt            45222 non-null int64
education         45222 non-null int8
education_num     45222 non-null int64
marital_status    45222 non-null int8
occupation        45222 non-null int8
relationship      45222 non-null int8
race              45222 non-null int8
sex               45222 non-null int8
capital_gain      45222 non-null int64
capital_loss      45222 non-null int64
hours_per_week    45222 non-null int64
native_country    45222 non-null int8
wage_class        45222 non-null int8
dtypes: int64(6), int8(9)
memory usage: 2.8 MB
```

Out[9]:

	age	workingclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	cap
0	39	5	77516	9	13	4	0	1	4	1	
1	50	4	83311	9	13	2	3	0	4	1	
2	38	2	215646	11	9	0	5	1	4	1	
3	53	2	234721	1	7	2	5	0	2	1	
4	28	2	338409	9	13	2	9	5	2	0	

In [10]:
```
#Split the train and test sets into their new respective dataframes
final_train=combined_set[:train_nomiss.shape[0]]
```

In [11]:
```
#setting up for XGB. model based on wage class.
y_train=final_train.pop('wage_class')
y_test=final_test.pop('wage_class')
y_train.head()
y_test.head()
```

Out[11]:
```
array([0, 1], dtype=int64)
```

In [12]:
```
#set up parameters for XGBoost, cv= cross validation, ind= index, GBM = gradient boost

cv_params1={'max_depth':[3,5,7], 'min_child_weight':[1,3,5]}

ind_params1={'learning_rate': 0.1 , 'n_estimators':1000, 'seed':123, 'subsample':0.8,
```

In [13]:
```
#create the model..
optimized_GBM= GridSearchCV(xgb.XGBClassifier(**ind_params1), cv_params1, scoring='acc
```

In [14]:
```
#run grid search with 5 fold cross validation, see which params perform the best.
```

Out[14]:
```
GridSearchCV(cv=5, error_score='raise',
       estimator=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel
=1,
       colsample_bytree=0.8, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=1000,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=123,
       silent=True, subsample=0.8),
       fit_params=None, iid=True, n_jobs=-1,
       param_grid={'max_depth': [3, 5, 7], 'min_child_weight': [1, 3, 5]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='accuracy', verbose=0)
```

In [15]:
```

```

```
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:761: De
precationWarning: The grid_scores_ attribute was deprecated in version 0.18 in fav
or of the more elaborate cv_results_ attribute. The grid_scores_ attribute will no
t be available from 0.20
  DeprecationWarning)
```

Out[15]:
```
[mean: 0.86761, std: 0.00292, params: {'max_depth': 3, 'min_child_weight': 1},
 mean: 0.86752, std: 0.00230, params: {'max_depth': 3, 'min_child_weight': 3},
 mean: 0.86778, std: 0.00292, params: {'max_depth': 3, 'min_child_weight': 5},
 mean: 0.86181, std: 0.00310, params: {'max_depth': 5, 'min_child_weight': 1},
 mean: 0.86181, std: 0.00262, params: {'max_depth': 5, 'min_child_weight': 3},
 mean: 0.86254, std: 0.00243, params: {'max_depth': 5, 'min_child_weight': 5},
 mean: 0.85744, std: 0.00373, params: {'max_depth': 7, 'min_child_weight': 1},
 mean: 0.85654, std: 0.00336, params: {'max_depth': 7, 'min_child_weight': 3},
 mean: 0.85754, std: 0.00425, params: {'max_depth': 7, 'min_child_weight': 5}]
```

In [16]:
```
#apply on subsampling params as well

cv_params2={'learning_rate': [0.1,0.01], 'subsample': [0.7,0.8,0.9]}
ind_params2={'n_estimatiors':1000, 'seed':123, 'colsample_bytree':0.8, 'objective':'bi

optimized_GBM2 = GridSearchCV(xgb.XGBClassifier(**ind_params2),
                              cv_params2,
                               scoring = 'accuracy', cv = 5, n_jobs = -1)
optimized_GBM2.fit(final_train, y_train)

optimized_GBM2.grid_scores_
```

```
C:\Users\HP\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:761: De
precationWarning: The grid_scores_ attribute was deprecated in version 0.18 in fav
or of the more elaborate cv_results_ attribute. The grid_scores_ attribute will no
t be available from 0.20
  DeprecationWarning)
```

Out[16]:
```
[mean: 0.86072, std: 0.00384, params: {'learning_rate': 0.1, 'subsample': 0.7},
 mean: 0.86006, std: 0.00386, params: {'learning_rate': 0.1, 'subsample': 0.8},
 mean: 0.86059, std: 0.00326, params: {'learning_rate': 0.1, 'subsample': 0.9},
 mean: 0.84142, std: 0.00514, params: {'learning_rate': 0.01, 'subsample': 0.7},
 mean: 0.84139, std: 0.00485, params: {'learning_rate': 0.01, 'subsample': 0.8},
 mean: 0.84119, std: 0.00491, params: {'learning_rate': 0.01, 'subsample': 0.9}]
```

In [17]:
```
#params are max_depth of 3, min_child_weight of 5, learning_rate:0.1, subsample:0.7


xgdmat=xgb.DMatrix(final_train,y_train)#create DMatrix for XGB

xgb_params={'eta':0.1, 'seed':123, 'subsample':0.7, 'colsample_bytree': 0.8, 'objectiv
```

In [18]:
```
cv_xgb=xgb.cv(params=xgb_params, dtrain=xgdmat, num_boost_round=3000, nfold=5, metrics

#look at the tail to see how accurate we got
print(cv_xgb.tail(5))
#got 88.3537% accurate!, took 542 rounds
```

```
[09:52:40] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:52:40] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:52:40] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 10 extra nodes, 0 pruned nodes, max_depth=3
[09:52:40] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 12 extra nodes, 0 pruned nodes, max_depth=3
[09:52:40] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 10 extra nodes, 0 pruned nodes, max_depth=3
[09:52:41] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:52:41] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:52:41] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 12 extra nodes, 0 pruned nodes, max_depth=3
[09:52:41] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:52:41] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
```

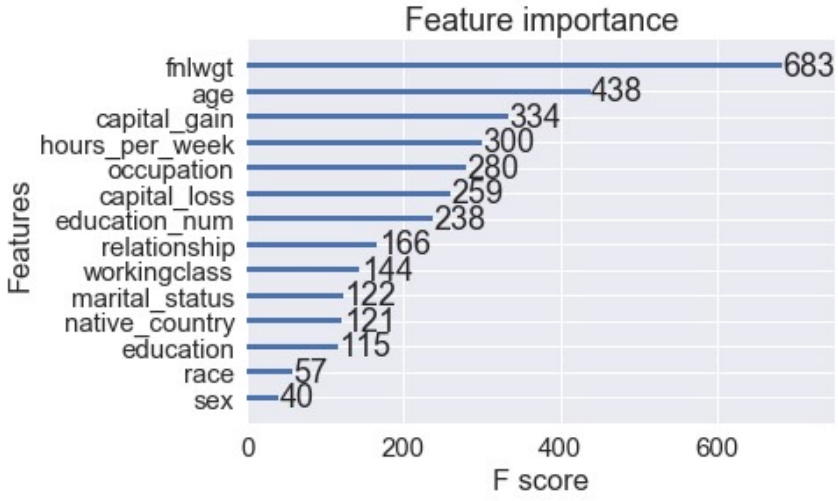
In [19]:
```
#create final callable model

final_params={'eta':0.1, 'seed':123, 'subsample':0.7, 'colsample_bytree': 0.8, 'object

final_gb=xgb.train(final_params, xgdmat, num_boost_round=542)
```

```
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 12 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 12 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
ee pruning end, 1 roots, 12 extra nodes, 0 pruned nodes, max_depth=3
[09:56:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tr
```

In [20]:
```
#use seaborn to plot
```

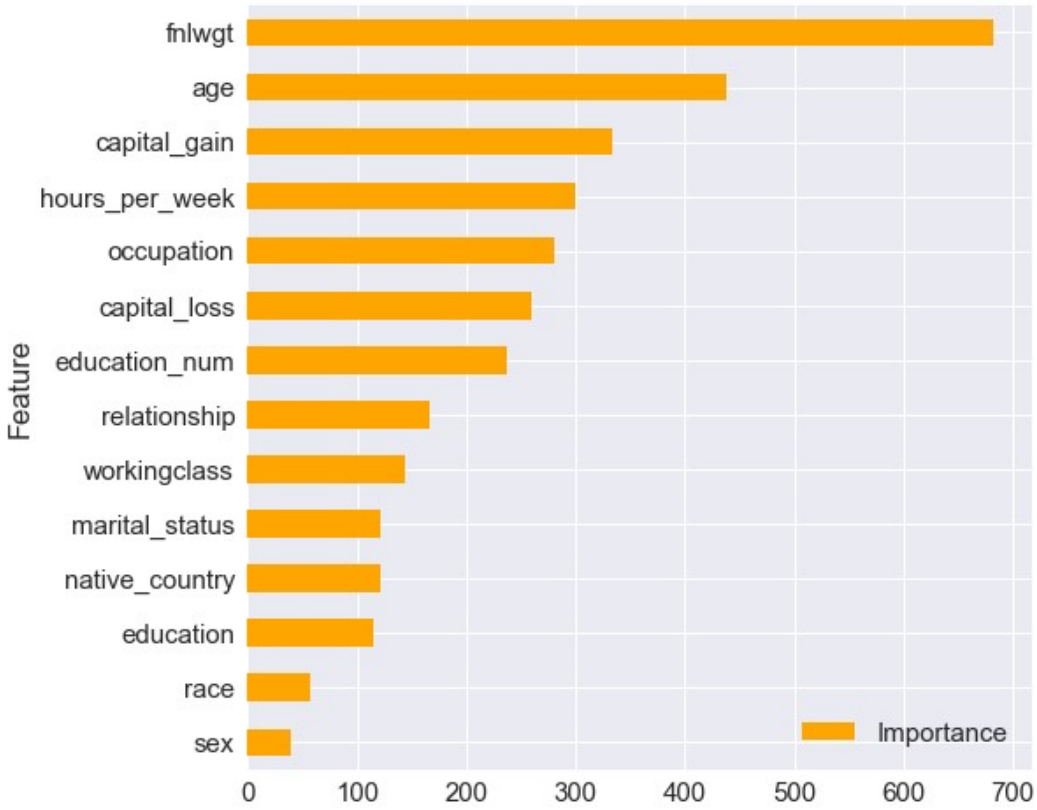
In [21]:
```
#plot feature importance. fnlwgt is most important with age and capital_gain coming se
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29305d416d8>
```



In [22]:
```
#make our own nice looking feature importance plot instead of using the builtin xgb.pl

importances = final_gb.get_fscore()
importances

importance_frame = pd.DataFrame({'Importance': list(importances.values()), 'Feature':
importance_frame.sort_values(by = 'Importance', inplace = True)
importance_frame.plot(kind = 'barh', x = 'Feature', figsize = (8,8), color = 'orange')
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2930765e908>
```



In [23]:
```
# model has now been tuned with cv grid search and early stopping. now test on test se

testdmat=xgb.DMatrix(final_test)
y_pred=final_gb.predict(testdmat)
```

Out[23]:
```
array([0.00261635, 0.18977417, 0.2550358 , ..., 0.8251352 , 0.15983744,
       0.7877533 ], dtype=float32)
```

In [24]:
```
#y_pred is outputted as probabilities by default, not class labels. To fix, we use:
y_pred=np.rint(y_pred)
```

In [25]:
```
accuracy_score(y_pred,y_test), 1-accuracy_score(y_pred,y_test)
```

Out[25]:
```
(0.8674634794156707, 0.1325365205843293)
```

In [ ]: