In [8]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import Imputer
from sklearn.naive_bayes import GaussianNB
```

In [9]:
```
import numpy as np
import xgboost as xgb
```

In [10]:
```
from numpy import loadtxt
from xgboost import XGBClassifier
```

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
adult_df = pd.read_table('https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data', sep='\s+', header=None)
adult_df.columns = ['age','workclass','fnlwgt','education','education_num','marital_status','occupation','relationship','race',
                    'sex', 'capital-gain','capital-loss','hours-per-week','native-country','income']
```

In [13]:
```
adult_df.head()
```

Out[13]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relat
0	39,	State-gov,	77516,	Bachelors,	13,	Never-married,	Adm-clerical,	Not-fami
1	50,	Self-emp-not-inc,	83311,	Bachelors,	13,	Married-civ-spouse,	Exec-managerial,	Husl
2	38,	Private,	215646,	HS-grad,	9,	Divorced,	Handlers-cleaners,	Not-fami
3	53,	Private,	234721,	11th,	7,	Married-civ-spouse,	Handlers-cleaners,	Husl
4	28,	Private,	338409,	Bachelors,	13,	Married-civ-spouse,	Prof-specialty,	Wife

In [14]:
```
data_ = pd.get_dummies(data = adult_df, columns=['workclass', 'educatio
n',  'marital_status', 'occupation', 'relationship', 'race', 'sex', 'na
tive-country'])
data_.head()
```

Out[14]:

	age	fnlwgt	education_num	capital-gain	capital-loss	hours-per-week	income	workclass_?,	workc
0	39,	77516,	13,	2174,	0,	40,	<=50K	0	0
1	50,	83311,	13,	0,	0,	13,	<=50K	0	0
2	38,	215646,	9,	0,	0,	40,	<=50K	0	0
3	53,	234721,	7,	0,	0,	40,	<=50K	0	0
4	28,	338409,	13,	0,	0,	40,	<=50K	0	0

5 rows × 109 columns

In [15]:
```
data_['fnlwgt'].unique()
```

Out[15]:
```
array(['77516,', '83311,', '215646,', ..., '34066,', '84661,', '257302
,'], dtype=object)
```

In [16]:
```
data_.columns.tolist()
```

Out[16]:
```
['age',
 'fnlwgt',
 'education_num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'income',
 'workclass_?,',
 'workclass_Federal-gov,',
 'workclass_Local-gov,',
 'workclass_Never-worked,',
 'workclass_Private,',
 'workclass_Self-emp-inc,',
 'workclass_Self-emp-not-inc,',
 'workclass_State-gov,',
 'workclass_Without-pay,',
 'education_10th,',
 'education_11th,',
 'education_12th,',
 'education_1st-4th,',
 'education_5th-6th,',
 'education_7th-8th,',
 'education_9th,',
 'education_Assoc-acdm,',
 'education_Assoc-voc,',
 'education_Bachelors,',
 'education_Doctorate,',
 'education_HS-grad,',
 'education_Masters,',
 'education_Preschool,',
 'education_Prof-school,',
 'education_Some-college,',
 'marital_status_Divorced,',
 'marital_status_Married-AF-spouse,',
 'marital_status_Married-civ-spouse,',
 'marital_status_Married-spouse-absent,',
 'marital_status_Never-married,',
 'marital_status_Separated,',
 'marital_status_Widowed,',
 'occupation_?,',
 'occupation_Adm-clerical,',
 'occupation_Armed-Forces,',
 'occupation_Craft-repair,',
 'occupation_Exec-managerial,',
 'occupation_Farming-fishing,',
 'occupation_Handlers-cleaners,',
 'occupation_Machine-op-inspct,',
 'occupation_Other-service,',
 'occupation_Priv-house-serv,',
 'occupation_Prof-specialty,',
```

```
'occupation_Protective-serv,',
'occupation_Sales,',
'occupation_Tech-support,',
'occupation_Transport-moving,',
'relationship_Husband,',
'relationship_Not-in-family,',
'relationship_Other-relative,',
'relationship_Own-child,',
'relationship_Unmarried,',
'relationship_Wife,',
'race_Amer-Indian-Eskimo,',
'race_Asian-Pac-Islander,',
'race_Black,',
'race_Other,',
'race_White,',
'sex_Female,',
'sex_Male,',
'native-country_?,',
'native-country_Cambodia,',
'native-country_Canada,',
'native-country_China,',
'native-country_Columbia,',
'native-country_Cuba,',
'native-country_Dominican-Republic,',
'native-country_Ecuador,',
'native-country_El-Salvador,',
'native-country_England,',
'native-country_France,',
'native-country_Germany,',
'native-country_Greece,',
'native-country_Guatemala,',
'native-country_Haiti,',
'native-country_Holand-Netherlands,',
'native-country_Honduras,',
'native-country_Hong,',
'native-country_Hungary,',
'native-country_India,',
'native-country_Iran,',
'native-country_Ireland,',
'native-country_Italy,',
'native-country_Jamaica,',
'native-country_Japan,',
'native-country_Laos,',
'native-country_Mexico,',
'native-country_Nicaragua,',
'native-country_Outlying-US(Guam-USVI-etc),',
'native-country_Peru,',
'native-country_Philippines,',
'native-country_Poland,',
'native-country_Portugal,',
'native-country_Puerto-Rico,',
'native-country_Scotland,',
```

```
 'native-country_South,',
 'native-country_Taiwan,',
 'native-country_Thailand,',
 'native-country_Trinadad&Tobago,',
 'native-country_United-States,',
 'native-country_Vietnam,',
 'native-country_Yugoslavia,']
```

In [17]:
```
data_['capital-gain'] = pd.to_numeric(data_['capital-gain'].str.replace(',',''))
data_['capital-loss'] = pd.to_numeric(data_['capital-loss'].str.replace(',',''))
data_['hours-per-week'] = pd.to_numeric(data_['hours-per-week'].str.replace(',',''))
data_['education_num'] = pd.to_numeric(data_['education_num'].str.replace(',',''))
data_['fnlwgt'] = pd.to_numeric(data_['fnlwgt'].str.replace(',',''))
data_['age'] = pd.to_numeric(data_['age'].str.replace(',',''))
```

In [18]:
```
data_['income'].replace({'>50K':'1', '<=50K':'0'}, inplace=True)
data_['income'] = pd.to_numeric(data_['income'].str.replace(',',''))
```

In [ ]:
```
data_.head()
```

In [19]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
X = data_[['age', 'fnlwgt', 'education_num', 'capital-gain', 'capital-l
oss', 'hours-per-week', 'workclass_?,', 'workclass_Federal-gov,', 'work
class_Local-gov,','workclass_Never-worked,', 'workclass_Private,', 'wor
kclass_Self-emp-inc,', 'workclass_Self-emp-not-inc,', 'workclass_State-
gov,', 'workclass_Without-pay,', 'education_10th,', 'education_11th,',
'education_12th,', 'education_1st-4th,', 'education_5th-6th,', 'educati
on_7th-8th,', 'education_9th,', 'education_Assoc-acdm,', 'education_Ass
oc-voc,', 'education_Bachelors,', 'education_Doctorate,', 'education_HS
-grad,', 'education_Masters,', 'education_Preschool,', 'education_Prof-
school,', 'education_Some-college,', 'marital_status_Divorced,', 'marit
al_status_Married-AF-spouse,', 'marital_status_Married-civ-spouse,', 'm
arital_status_Married-spouse-absent,', 'marital_status_Never-married,',
'marital_status_Separated,', 'marital_status_Widowed,', 'occupation_?,'
, 'occupation_Adm-clerical,', 'occupation_Armed-Forces,', 'occupation_C
raft-repair,', 'occupation_Exec-managerial,', 'occupation_Farming-fishi
ng,', 'occupation_Handlers-cleaners,', 'occupation_Machine-op-inspct,',
'occupation_Other-service,', 'occupation_Priv-house-serv,', 'occupation
_Prof-specialty,', 'occupation_Protective-serv,', 'occupation_Sales,',
'occupation_Tech-support,', 'occupation_Transport-moving,', 'relationsh
ip_Husband,', 'relationship_Not-in-family,', 'relationship_Other-relati
ve,', 'relationship_Own-child,', 'relationship_Unmarried,', 'relationsh
ip_Wife,', 'race_Amer-Indian-Eskimo,', 'race_Asian-Pac-Islander,', 'rac
e_Black,', 'race_Other,', 'race_White,', 'sex_Female,', 'sex_Male,', 'n
ative-country_?,', 'native-country_Cambodia,', 'native-country_Canada,'
, 'native-country_China,', 'native-country_Columbia,', 'native-country_
Cuba,', 'native-country_Dominican-Republic,', 'native-country_Ecuador,'
, 'native-country_El-Salvador,', 'native-country_England,', 'native-cou
ntry_France,', 'native-country_Germany,', 'native-country_Greece,', 'na
tive-country_Guatemala,', 'native-country_Haiti,', 'native-country_Hola
nd-Netherlands,', 'native-country_Honduras,', 'native-country_Hong,', '
native-country_Hungary,', 'native-country_India,', 'native-country_Iran
,', 'native-country_Ireland,', 'native-country_Italy,', 'native-country
_Jamaica,', 'native-country_Japan,', 'native-country_Laos,', 'native-co
untry_Mexico,', 'native-country_Nicaragua,', 'native-country_Outlying-U
S(Guam-USVI-etc),', 'native-country_Peru,', 'native-country_Philippines
,', 'native-country_Poland,', 'native-country_Portugal,', 'native-count
ry_Puerto-Rico,', 'native-country_Scotland,', 'native-country_South,',
'native-country_Taiwan,', 'native-country_Thailand,', 'native-country_T
rinadad&Tobago,', 'native-country_United-States,', 'native-country_Viet
nam,', 'native-country_Yugoslavia,']]
Y = data_['income']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3
,random_state =12312)
```

In [20]:
```
m1 = DecisionTreeClassifier()
m1.fit(X_train,Y_train)
```

Out[20]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', 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]:
```
train_DT_pred = m1.predict(X_train)
test_DT_pred = m1.predict(X_test)
```

In [22]:
```
print(accuracy_score(Y_train, train_DT_pred))
print(accuracy_score(Y_test, test_DT_pred))
```

```
1.0
0.816972054458
```

In [23]:
```
m2 = GaussianNB()
m2.fit(X_train,Y_train)
train_NB_pred = m2.predict(X_train)
test_NB_pred = m2.predict(X_test)
```

In [24]:
```
print(accuracy_score(Y_train, train_NB_pred))
print(accuracy_score(Y_test, test_NB_pred))
```

```
0.794357669358
0.79793223462
```

In [25]:
```
m3 = LogisticRegression()
m3.fit(X_train,Y_train)
train_LR_pred = m3.predict(X_train)
test_LR_pred = m3.predict(X_test)
```

In [26]:
```
print(accuracy_score(Y_train, train_LR_pred))
print(accuracy_score(Y_test, test_LR_pred))
```

```
0.795981045981
0.799877162453
```

In [27]:
```
m4 = XGBClassifier(n_estimators=5000)
m4.fit(X_train,Y_train)
train_GB_pred = m4.predict(X_train)
test_GB_pred = m4.predict(X_test)
```

In [28]:
```
print(accuracy_score(Y_train, train_GB_pred))
print(accuracy_score(Y_test, test_GB_pred))
```

```
0.934011934012
0.866721261132
```

In [29]:
```
data_.shape
```

Out[29]: (32561, 109)

In [30]:
```
m5 = KNeighborsClassifier(n_neighbors=5)
m5.fit(X_train, Y_train)
```

Out[30]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski
',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [31]:
```
train_KN_pred = m5.predict(X_train)
test_KN_pred = m5.predict(X_test)
```

In [32]:
```
print(accuracy_score(Y_train, train_KN_pred))
print(accuracy_score(Y_test, test_KN_pred))
```

```
0.831168831169
0.777049851571
```

In [33]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [34]:
```
m6 = RandomForestClassifier(n_estimators=70,oob_score=True,random_state
=45)
m6.fit(X_train, Y_train)
```

Out[34]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
ini',
            max_depth=None, max_features='auto', 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, n_estimators=70, n_jobs=1,
            oob_score=True, random_state=45, verbose=0, warm_start=Fal
se)
```

In [35]:
```
train_RF_pred = m6.predict(X_train)
test_RF_pred = m6.predict(X_test)
```

In [36]:
```
print(accuracy_score(Y_train, train_RF_pred))
print(accuracy_score(Y_test, test_RF_pred))
```

```
0.999605124605
0.856280069608
```

In [37]:
```
Ensemble_M_train = pd.DataFrame({'DT':list(train_DT_pred),'NB':list(tra
in_NB_pred),'LR':list(train_LR_pred), 'GB': list(train_GB_pred), 'KN':
list(train_KN_pred), 'RF': list(train_RF_pred), 'Y': list(Y_train)})
Ensemble_M_test = pd.DataFrame({'DT':list(test_DT_pred),'NB':list(test_
NB_pred),'LR':list(test_LR_pred), 'GB': list(test_GB_pred), 'KN': list(
test_KN_pred), 'RF': list(test_RF_pred), 'Y': list(Y_test)})
```

In [38]:
```
Ensemble_M_train.head()
```

Out[38]:

	DT	GB	KN	LR	NB	RF	Y
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

In [39]:
```
Ensemble_M_test.head()
```

Out[39]:

	DT	GB	KN	LR	NB	RF	Y
0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	1	1	0	1	1	1	1
4	0	0	0	0	0	0	0

In [40]:
```
pd.crosstab(Ensemble_M_test.LR, Ensemble_M_test.Y)
```

Out[40]:

Y	0	1
LR		
0	7211	1698
1	257	603

In [41]:
```
X_train_Ens = Ensemble_M_train[['DT','NB','LR','GB','KN','RF']]
Y_train_Ens = Ensemble_M_train[['Y']]
```

In [42]:
```
X_test_Ens = Ensemble_M_test[['DT','NB','LR','GB','KN','RF']]
Y_test_Ens = Ensemble_M_test[['Y']]
```

In [43]:
```
model_ensemble = LogisticRegression()
```

In [44]:
```
model_ensemble.fit(X_train_Ens, Y_train_Ens)
```

```
/anaconda/lib/python3.5/site-packages/sklearn/utils/validation.py:547:
DataConversionWarning: A column-vector y was passed when a 1d array wa
s expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
```

Out[44]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0
001,
          verbose=0, warm_start=False)
```

In [45]:
```
test_pred_Ens = model_ensemble.predict(X_test_Ens)
```

In [46]:
```
print(classification_report(Y_test_Ens, test_pred_Ens))
```

```
             precision    recall  f1-score   support

          0       0.89      0.87      0.88      7468
          1       0.61      0.64      0.62      2301

avg / total       0.82      0.82      0.82      9769
```

In [47]:
```
print(accuracy_score(Y_test_Ens, test_pred_Ens))
```

```
0.816972054458
```

In [48]:
```
print(confusion_matrix(Y_test_Ens, test_pred_Ens))
```

```
[[6507  961]
 [ 827 1474]]
```

In [49]:
```
tn, fp, fn, tp = confusion_matrix(Y_test_Ens, test_pred_Ens).ravel()
```

In [50]:
```
tn, fp, fn, tp
```

Out[50]:
```
(6507, 961, 827, 1474)
```

In [51]:
```
import matplotlib.pyplot as plt

%matplotlib inline
```

In [52]:
```
adult_df['education_num'] = pd.to_numeric(adult_df['education_num'].str
.replace(',',''))
```

In [55]:
```
import ggplot
from ggplot import *
```

```
---------------------------------------------------------------------
-----
ImportError                               Traceback (most recent call
last)
<ipython-input-55-d5ae4e502d84> in <module>()
----> 1 import ggplot
      2 from ggplot import *

ImportError: No module named 'ggplot'
```

In [56]:
```
adult_df['age'] = pd.to_numeric(adult_df['age'].str.replace(',',''))
```

In [57]:
```
adult_df.head()
```

Out[57]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relat
0	39	State-gov,	77516,	Bachelors,	13	Never-married,	Adm-clerical,	Not-fami
1	50	Self-emp-not-inc,	83311,	Bachelors,	13	Married-civ-spouse,	Exec-managerial,	Husl
2	38	Private,	215646,	HS-grad,	9	Divorced,	Handlers-cleaners,	Not-fami
3	53	Private,	234721,	11th,	7	Married-civ-spouse,	Handlers-cleaners,	Husl
4	28	Private,	338409,	Bachelors,	13	Married-civ-spouse,	Prof-specialty,	Wife

In [ ]:
```
ggplot(aes(x='age', fill='income'), data=adult_df) + \
    geom_histogram(color='black')
```

In [ ]:
```
ggplot(aes(x='education_num', fill='income'), adult_df) + geom_histogra
m(color='black')
```

In [ ]:
```
adult_df.occupation.unique()
```

In [ ]:
```
adult_df.occupation.replace({'Adm-clerical':'AC', 'Exec-managerial':'EM
', 'Handlers-cleaners':'HC',
       'Prof-specialty':'PS', 'Other-service':'OS', 'Sales':'S', 'Craft
-repair':'CR',
       'Transport-moving':'TM', 'Farming-fishing':'FF', 'Machine-op-ins
pct':'MOI',
       'Tech-support':'TS', '?':'?', 'Protective-serv':'PSE', 'Armed-Fo
rces':'AF',
       'Priv-house-serv':'PHS'}, inplace= True)
```

In [ ]:
```
adult_df.income = adult_df.income.apply(lambda x:1 if x == '>50K' else
0)
adult_df.income.unique()
```

In [ ]:
```
fig, axs = plt.subplots(figsize=(18, 10), sharey=True)
import seaborn as sns
sns.barplot(x="occupation", y="income", data=adult_df);
```

In [ ]:
```
sns.barplot(x="race", y="income", data=adult_df);
```

In [ ]:
```
for i in range(len(adult_df['hours-per-week'])):
    adult_df['hours-per-week'][i] = adult_df['hours-per-week'][i][:len(
adult_df['hours-per-week'][i])-1]
```

In [ ]:
```
fig, ax = plt.subplots(1,1,figsize=(18,20))
m4.feature_importances_
xgb.plot_importance(m4, ax=ax, max_num_features=106)
```

In [ ]:
```
fig, ax = plt.subplots(1,1,figsize=(18,20))
m4.feature_importances_
xgb.plot_importance(m4, ax=ax, max_num_features=106)
```