Assignment 25

Problem Statement

In this assignment students need to predict whether a person makes over 50K per year or not from classic adult dataset using XGBoost.

The description of the dataset is as follows:

Data Set Information:

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Attribute Information:

- (>50K, <=50K)
- age: continuous
- workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
- fnlwgt: continuous
- education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
- education-num: continuous.
- marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
- sex: Female, Male
- capital-gain: continuous
- capital-loss: continuous
- hours-per-week: continuous
- native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands

In [136]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
import xgboost as xgb
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
```

In [96]:
```
# Load the training dataset from the URL
train_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data', header = None)
```

In [97]:
```
# Load the test data set from the URL
test_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test', skiprows = 1, h
eader = None)
```

In [98]:
```
# Display the top five rows of the training dataset
train_set.head()
```

Out[98]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [99]:
```
# Display the top five rows of the test dataset
test_set.head()
```

Out[99]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K.
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K.
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K.
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K.
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K.

In [100]:
```
# I am going to combine the two datasets together and split them later when I build the Machine Learning Model
dataset = pd.concat([train_set, test_set])
```

In [101]:
```
# The dataset does not have descriptive labels for the columns, therefore, I will add the labels
col_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation','relations
hip',
              'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'wage_class']
```

In [102]:
```
# Add the labels
dataset.columns = col_labels
```

In [103]:
```
dataset.head()
```

Out[103]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hour
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	

In [158]:
```
# The number of records for the dataset after combining.
dataset.shape
```

Out[158]: (48842, 15)

In [105]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48842 entries, 0 to 16280
Data columns (total 15 columns):
age               48842 non-null int64
workclass         48842 non-null object
fnlwgt            48842 non-null int64
education         48842 non-null object
education_num     48842 non-null int64
marital_status    48842 non-null object
occupation        48842 non-null object
relationship      48842 non-null object
race              48842 non-null object
sex               48842 non-null object
capital_gain      48842 non-null int64
capital_loss      48842 non-null int64
hours_per_week    48842 non-null int64
native_country    48842 non-null object
wage_class        48842 non-null object
dtypes: int64(6), object(9)
memory usage: 6.0+ MB
```

In [106]:
```
# See the statistics of the numeric columns
dataset.describe()
```

Out[106]:

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

In [107]:
```
# Count the number of null values in the columns
dataset.isnull().sum(axis=0)
```

Out[107]:
```
age               0
workclass         0
fnlwgt            0
education         0
education_num     0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
wage_class        0
dtype: int64
```

Exploratory Data Analysis

See the anomolies, patterns, trends, and relationship in our dataset.

In [108]:
```
# Count the number of people by Native Country.
pd.value_counts(dataset['native_country'])
```

Out[108]:
```
United-States                 43832
Mexico                          951
?                               857
Philippines                     295
Germany                         206
Puerto-Rico                     184
Canada                          182
El-Salvador                     155
India                           151
Cuba                            138
England                         127
China                           122
South                           115
Jamaica                         106
Italy                           105
Dominican-Republic              103
Japan                            92
Guatemala                        88
Poland                           87
Vietnam                          86
Columbia                         85
Haiti                            75
Portugal                         67
Taiwan                           65
Iran                             59
Greece                           49
Nicaragua                        49
Peru                             46
Ecuador                          45
France                           38
Ireland                          37
Hong                             30
Thailand                         30
Cambodia                         28
Trinadad&Tobago                  27
Yugoslavia                       23
Laos                             23
Outlying-US(Guam-USVI-etc)       23
Scotland                         21
Honduras                         20
Hungary                          19
Holand-Netherlands                1
Name: native_country, dtype: int64
```

In [109]:
```
# I tried replacing the ? in Native Country column with Unknown and couldn't. See if there are whitespaces in the d
ataset.

white_space = dataset['native_country'].str.isalpha()
```

In [110]:
```
print(white_space.head(20))
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
Name: native_country, dtype: bool
```

In [111]:
```
# Apparently the data in the native column has whitepce. Just to be cautious I will strip the white space from all
# the non numeric columns

dataset = dataset.apply(lambda x: x.str.strip() if x.dtype == "object" else x)
```

In [112]:
```
# I am going to replace ? in the Native Country with Unknown
dataset['native_country'].replace('?','Unknown', inplace=True)
```

In [113]:
```
pd.value_counts(dataset['native_country'])
```

Out[113]:
```
United-States                 43832
Mexico                          951
Unknown                         857
Philippines                     295
Germany                         206
Puerto-Rico                     184
Canada                          182
El-Salvador                     155
India                           151
Cuba                            138
England                         127
China                           122
South                           115
Jamaica                         106
Italy                           105
Dominican-Republic              103
Japan                            92
Guatemala                        88
Poland                           87
Vietnam                          86
Columbia                         85
Haiti                            75
Portugal                         67
Taiwan                           65
Iran                             59
Nicaragua                        49
Greece                           49
Peru                             46
Ecuador                          45
France                           38
Ireland                          37
Hong                             30
Thailand                         30
Cambodia                         28
Trinadad&Tobago                  27
Outlying-US(Guam-USVI-etc)       23
Laos                             23
Yugoslavia                       23
Scotland                         21
Honduras                         20
Hungary                          19
Holand-Netherlands                1
Name: native_country, dtype: int64
```

In [114]:
```
pd.value_counts(dataset['wage_class'])
```

Out[114]:
```
<=50K     24720
<=50K.    12435
>50K       7841
>50K.      3846
Name: wage_class, dtype: int64
```

In [115]:
```
# Looks like the wage class column has values with a period at the end.
# Replace those so there is no error when building my model.
dataset['wage_class'].replace(['<=50K.','>50K.'],['<=50K','>50K'], inplace=True)
```

In [116]:
```
pd.value_counts(dataset['wage_class'])
```

Out[116]:
```
<=50K    37155
>50K     11687
Name: wage_class, dtype: int64
```

In [117]:
```
# Replace the wage class column with 1 if person makes more than 50K and 0 if less than or equal to 50K
dataset['wage_class'].replace(['<=50K','>50K'],[0,1], inplace=True)
```

In [118]:
```
pd.value_counts(dataset['wage_class'])
```

Out[118]:
```
0    37155
1    11687
Name: wage_class, dtype: int64
```

In [119]:
```
pd.value_counts(dataset['workclass'])
```

Out[119]:
```
Private             33906
Self-emp-not-inc     3862
Local-gov            3136
?                    2799
State-gov            1981
Self-emp-inc         1695
Federal-gov          1432
Without-pay            21
Never-worked           10
Name: workclass, dtype: int64
```

In [120]:
```
# Replace ? with Unknown in work class column
dataset['workclass'].replace('?','Unknown', inplace=True)
```

In [121]:
```
dataset.isnull().sum(axis=0)
```

Out[121]:
```
age               0
workclass         0
fnlwgt            0
education         0
education_num     0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
wage_class        0
dtype: int64
```

In [122]:
```
pd.value_counts(dataset['marital_status'])
```

Out[122]:
```
Married-civ-spouse       22379
Never-married            16117
Divorced                  6633
Separated                 1530
Widowed                   1518
Married-spouse-absent      628
Married-AF-spouse           37
Name: marital_status, dtype: int64
```

Building the XGBoost ML Model

Split the dataset into training set and test set

In [124]:
```
# Split the feature set and target set

features = dataset.drop(columns='wage_class')
targets = dataset['wage_class']
```

In [126]:
```
# Label and Onehot encode the categorical values in features set

features = pd.get_dummies(features,drop_first=True)
```

In [127]:
```
features.head()
```

Out[127]:

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_em
0	39	77516	13	2174	0	40	0	0	0	
1	50	83311	13	0	0	13	0	0	0	
2	38	215646	9	0	0	40	0	0	1	
3	53	234721	7	0	0	40	0	0	1	
4	28	338409	13	0	0	40	0	0	1	

5 rows × 100 columns

In [133]:
```
# Split the data into 70% training set and 30% test set

X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size = 0.30, random_state = 0)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(34189, 100)
(14653, 100)
(34189,)
(14653,)
```

Create the Model

In [152]:
```
xgb_model = xgb.XGBClassifier()
xgb_model.fit(X_train,y_train)
```

Out[152]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

In [153]:
```
# Make predictions
y_pred = xgb_model.predict(X_test)
```

In [154]:
```
conf_mat = confusion_matrix(y_test,y_pred)
print(conf_mat)
```

```
[[10611   589]
 [ 1426  2027]]
```

In [155]:
```
# Evaluate the model performance metric

accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

```
0.8624854978502696
```

In [180]:
```
# Extract the feature importances into a dataframe
feature_results = pd.DataFrame({'feature': list(features.columns),
                                'importance': xgb_model.feature_importances_})

# Show the top 20 most important features of our dataset
feature_results = feature_results.sort_values('importance', ascending = False).reset_index(drop=True)

print(feature_results.head(20))
```

```
                            feature  importance
0                               age    0.153179
1                      capital_gain    0.151734
2                      capital_loss    0.135838
3                     education_num    0.117052
4                    hours_per_week    0.088150
5   marital_status_Married-civ-spouse    0.057803
6                            fnlwgt    0.037572
7                 relationship_Wife    0.028902
8        workclass_Self-emp-not-inc    0.026012
9            relationship_Unmarried    0.021676
10       occupation_Exec-managerial    0.021676
11         occupation_Other-service    0.015896
12                         sex_Male    0.014451
13       occupation_Farming-fishing    0.014451
14           relationship_Own-child    0.011561
15        occupation_Prof-specialty    0.011561
16              workclass_Local-gov    0.010116
17            native_country_Mexico    0.008671
18     occupation_Handlers-cleaners    0.008671
19          occupation_Tech-support    0.007225
```

In [173]:
```
# Using the given Census data, our XGBoost Machine Learning Model can predict if a person makes above 50K
# or at or below 50K with an 86% accuracy.
```

In [ ]: