Tubes2A_13515011

November 19, 2017

```
In [1]: import pandas as pd

        #PEMBACAAN DATASET CENSUS
        X = pd.read_csv('DatasetEskperimen/CensusIncome/CencusIncome.data.txt', sep=",", heade
        print("Overview data:")
        print(X.head())
        target = X["50K"]
        print("\n\nTARGET: ")
        print(target.head())

        census = X[["age", "workclass", "fnlwgt","education","education-num","marital-status","


        print("\n\nDATA: ")
        print(census.head())
        print()
```

```
Overview data:
   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    50K
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
```

```
TARGET:
0    <=50K
1    <=50K
2    <=50K
3    <=50K
4    <=50K
Name: 50K, dtype: object


DATA:
   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
0          2174             0              40   United-States
1             0             0              13   United-States
2             0             0              40   United-States
3             0             0              40   United-States
4             0             0              40            Cuba
```

In [2]:

```
#Training dengan KNN , kFold 10 fold , metrics, confusion matrix
import numpy as np
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
from sklearn.datasets import load_svmlight_files
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

```
#iris = datasets.load_iris()

#PEMBACAAN DATASET CENSUS
cen = pd.read_csv('DatasetEskperimen/CensusIncome/CencusIncome.data.txt', sep=",", head

census_data = cen[["age", "workclass", "fnlwgt","education","education-num","marital-st
census_target = cen["50K"]


#changing target into float 0 and 1
new = []

for index, item in enumerate(census_target):
    if (item == " <=50K"):
        new.append(0.0)
    else:
        if(item == " >50K"):
            new.append(1.0)
        else:
            new.append(2.0)

new = np.array(new)

new_data = pd.get_dummies(census_data)
new_data = new_data.values


split_number = 10

#folding
kf = KFold(n_splits=split_number,shuffle= True)
test = kf.split(new_data)
jumlah = 0;
nomorFold = 1
for train_index,test_index in test:
    data_train,data_test = new_data[train_index],new_data[test_index]
    target_train,target_test = new[train_index], new[test_index]

    #learning dataset
    knn = KNeighborsClassifier(n_neighbors = 3)
    knn.fit(data_train,target_train)
    print("fold ke: ",nomorFold)
    #predicting learning data
    prediction = knn.predict(data_test)
    print('PREDICTION: ',prediction)
    print('TARGET TEST : ',target_test)

    #generating confusion matrix
```

```
            #conf_matrix =
            conf = confusion_matrix(target_test,prediction)
            print('Confusion Matrix:')
            print(conf)

            #accuracy
            print('\nAccuracy:')
            acc = accuracy_score(target_test,prediction)
            jumlah+=acc
            print(acc*100, "%")

            #precision
            print('\nPrecission:')
            prec = precision_score(target_test,prediction)
            print(prec)

            #recall
            print('\nRecall:')
            rec = recall_score(target_test,prediction)
            print(rec)
            print('\n')
            nomorFold+=1

        average = jumlah/10;
        print('Rata-rata accuracy: ',average*100,'%\n')
```

```
fold ke:  1
PREDICTION:  [ 0.  0.  0. ...,  0.  1.  0.]
TARGET TEST :  [ 0.  0.  1. ...,  0.  1.  0.]
Confusion Matrix:
[[2196  302]
 [ 448  311]]

Accuracy:
76.9726742401 %

Precission:
0.507340946166

Recall:
0.409749670619


fold ke:  2
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  1. ...,  0.  0.  0.]
Confusion Matrix:
[[2173  296]
```

```
 [ 481  306]]

Accuracy:
76.1363636364 %

Precission:
0.508305647841

Recall:
0.388818297332


fold ke:  3
PREDICTION:  [ 0.  0.  0. ...,  1.  0.  0.]
TARGET TEST :  [ 1.  0.  0. ...,  1.  1.  0.]
Confusion Matrix:
[[2171  287]
 [ 484  314]]

Accuracy:
76.3206388206 %

Precission:
0.522462562396

Recall:
0.393483709273


fold ke:  4
PREDICTION:  [ 1.  1.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  1.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2212  304]
 [ 462  278]]

Accuracy:
76.4742014742 %

Precission:
0.477663230241

Recall:
0.375675675676


fold ke:  5
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
```

```
TARGET TEST :  [ 1.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2206  260]
 [ 495  295]]

Accuracy:
76.812039312 %

Precission:
0.531531531532

Recall:
0.373417721519


fold ke:  6
PREDICTION:  [ 0.  0.  1. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2188  266]
 [ 493  309]]

Accuracy:
76.6891891892 %

Precission:
0.537391304348

Recall:
0.385286783042


fold ke:  7
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 1.  1.  0. ...,  1.  0.  0.]
Confusion Matrix:
[[2149  317]
 [ 516  274]]

Accuracy:
74.4164619165 %

Precission:
0.463620981387

Recall:
0.346835443038
```

```
fold ke:  8
PREDICTION:  [ 0.  1.  0. ...,  0.  1.  1.]
TARGET TEST :  [ 0.  1.  1. ...,  0.  1.  1.]
Confusion Matrix:
[[2192  270]
 [ 517  277]]

Accuracy:
75.8292383292 %

Precission:
0.506398537477

Recall:
0.348866498741


fold ke:  9
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  1.  0.  0.]
Confusion Matrix:
[[2194  308]
 [ 472  282]]

Accuracy:
76.0442260442 %

Precission:
0.477966101695

Recall:
0.37400530504


fold ke:  10
PREDICTION:  [ 0.  0.  0. ...,  0.  1.  0.]
TARGET TEST :  [ 0.  0.  1. ...,  0.  0.  0.]
Confusion Matrix:
[[2125  304]
 [ 502  325]]

Accuracy:
75.2457002457 %

Precission:
0.516693163752
```

```
Recall:
0.392986698912


Rata-rata accuracy:  76.0940733208 %
```

In [3]:
```python
#Training dengan Naive Bayes , kFold 10 fold , metrics, confusion matrix
import numpy as np
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
import pandas as pd
from sklearn.datasets import load_svmlight_files
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

#iris = datasets.load_iris()

#PEMBACAAN DATASET CENSUS
cen = pd.read_csv('DatasetEskperimen/CensusIncome/CencusIncome.data.txt', sep=",", head

census_data = cen[["age", "workclass", "fnlwgt","education","education-num","marital-s
census_target = cen["50K"]


#changing target into float 0 and 1
new = []

for index, item in enumerate(census_target):
    if (item == " <=50K"):
        new.append(0.0)
    else:
        if(item == " >50K"):
            new.append(1.0)
        else:
            new.append(2.0)

new = np.array(new)

new_data = pd.get_dummies(census_data)
new_data = new_data.values
```

```
split_number = 10

#folding
kf = KFold(n_splits=split_number,shuffle= True)
jumlah = 0
nomorFold = 1
for train_index,test_index in kf.split(new_data):
    data_train,data_test = new_data[train_index],new_data[test_index]
    target_train,target_test = new[train_index], new[test_index]

    #learning dataset
    gnb = GaussianNB()
    gnb.fit(data_train,target_train)
    print("fold ke: ",nomorFold)
    #predicting learning data
    prediction = gnb.predict(data_test)
    print('PREDICTION: ',prediction)
    print('TARGET TEST : ',target_test)

    #generating confusion matrix
    #conf_matrix =
    conf = confusion_matrix(target_test,prediction)
    print('Confusion Matrix:')
    print(conf)

    #accuracy
    print('\nAccuracy:')
    acc = accuracy_score(target_test,prediction)
    jumlah+=acc
    print(acc*100,'%')

    #precision
    print('\nPrecission:')
    prec = precision_score(target_test,prediction)
    print(prec)

    #recall
    print('\nRecall:')
    rec = recall_score(target_test,prediction)
    print(rec)
    print('\n')
    nomorFold+=1

average = jumlah/10;
print('Rata-rata accuracy: ',average*100,'%\n')
```

```
fold ke:  1
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
```

```
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2335  137]
 [ 554  231]]

Accuracy:
78.7841571999 %

Precission:
0.627717391304

Recall:
0.294267515924


fold ke:  2
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 1.  1.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2354  135]
 [ 512  255]]

Accuracy:
80.128992629 %

Precission:
0.653846153846

Recall:
0.332464146023


fold ke:  3
PREDICTION:  [ 1.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 1.  1.  0. ...,  0.  1.  0.]
Confusion Matrix:
[[2323  125]
 [ 571  237]]

Accuracy:
78.6240786241 %

Precission:
0.654696132597

Recall:
0.293316831683
```

```
fold ke:  4
PREDICTION:  [ 0.  1.  0. ...,  0.  0.  1.]
TARGET TEST :  [ 0.  1.  0. ...,  1.  0.  1.]
Confusion Matrix:
[[2302  139]
 [ 582  233]]

Accuracy:
77.8562653563 %

Precission:
0.626344086022

Recall:
0.285889570552


fold ke:  5
PREDICTION:  [ 0.  0.  1. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  1.]
Confusion Matrix:
[[2378  114]
 [ 523  241]]

Accuracy:
80.4361179361 %

Precission:
0.678873239437

Recall:
0.315445026178


fold ke:  6
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 1.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2361  118]
 [ 525  252]]

Accuracy:
80.2518427518 %

Precission:
0.681081081081
```

```
Recall:
0.324324324324


fold ke:  7
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 1.  1.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2333  128]
 [ 537  258]]

Accuracy:
79.5761670762 %

Precission:
0.668393782383

Recall:
0.324528301887


fold ke:  8
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  1.  0.  1.]
Confusion Matrix:
[[2328  123]
 [ 538  267]]

Accuracy:
79.699017199 %

Precission:
0.684615384615

Recall:
0.331677018634


fold ke:  9
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  1.  0.]
Confusion Matrix:
[[2368  126]
 [ 535  227]]

Accuracy:
79.699017199 %
```

```
Precission:
0.643059490085

Recall:
0.297900262467


fold ke:  10
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2379  114]
 [ 528  235]]

Accuracy:
80.2825552826 %

Precission:
0.67335243553

Recall:
0.307994757536


Rata-rata accuracy:  79.5338211254 %
```

In [4]:
```
#Training dengan Decision Tree , kFold 10 fold , metrics, confusion matrix
import numpy as np
from sklearn import datasets
from sklearn import tree
import pandas as pd
from sklearn.datasets import load_svmlight_files
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

#PEMBACAAN DATASET CENSUS
cen = pd.read_csv('DatasetEskperimen/CensusIncome/CencusIncome.data.txt', sep=",", head

census_data = cen[["age", "workclass", "fnlwgt","education","education-num","marital-s
census_target = cen["50K"]
```

```python
#changing target into float 0 and 1
new = []

for index, item in enumerate(census_target):
    if (item == " <=50K"):
        new.append(0.0)
    else:
        if(item == " >50K"):
            new.append(1.0)
        else:
            new.append(2.0)

new = np.array(new)

new_data = pd.get_dummies(census_data)
new_data = new_data.values


split_number = 10

#folding
kf = KFold(n_splits=split_number,shuffle= True)
jumlah=0
nomorFold = 1
for train_index,test_index in kf.split(new_data):
    data_train,data_test = new_data[train_index],new_data[test_index]
    target_train,target_test = new[train_index], new[test_index]

    #learning dataset
    clf = tree.DecisionTreeClassifier()
    clf.fit(data_train,target_train)
    print("fold ke ",nomorFold)
    #predicting learning data
    prediction = clf.predict(data_test)
    print('PREDICTION: ',prediction)
    print('TARGET TEST : ',target_test)

    #generating confusion matrix
    #conf_matrix =
    conf = confusion_matrix(target_test,prediction)
    print('Confusion Matrix:')
    print(conf)

    #accuracy
    print('\nAccuracy:')
    acc = accuracy_score(target_test,prediction)
    jumlah+=acc
    print(acc*100,'%')
```

```
            #precision
            print('\nPrecission:')
            prec = precision_score(target_test,prediction)
            print(prec)

            #recall
            print('\nRecall:')
            rec = recall_score(target_test,prediction)
            print(rec)
            print('\n')
            nomorFold+=1

        average = jumlah/10;
        print('Rata-rata accuracy: ',average*100,'%\n')
```

```
fold ke  1
PREDICTION:  [ 1.  0.  0. ...,  0.  0.  1.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  1.]
Confusion Matrix:
[[2167  316]
 [ 272  502]]

Accuracy:
81.9465766042 %

Precission:
0.61369193154

Recall:
0.64857881137


fold ke  2
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  1.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  1.]
Confusion Matrix:
[[2192  299]
 [ 283  482]]

Accuracy:
82.1253071253 %

Precission:
0.617157490397

Recall:
0.630065359477
```

```
fold ke  3
PREDICTION:  [ 0.  0.  1. ...,  0.  0.  1.]
TARGET TEST :  [ 0.  0.  1. ...,  0.  0.  1.]
Confusion Matrix:
[[2155  348]
 [ 262  491]]

Accuracy:
81.2653562654 %

Precission:
0.585220500596

Recall:
0.652058432935


fold ke  4
PREDICTION:  [ 1.  0.  0. ...,  0.  1.  0.]
TARGET TEST :  [ 1.  0.  0. ...,  0.  1.  0.]
Confusion Matrix:
[[2162  318]
 [ 289  487]]

Accuracy:
81.3574938575 %

Precission:
0.604968944099

Recall:
0.627577319588


fold ke  5
PREDICTION:  [ 0.  1.  1. ...,  0.  1.  0.]
TARGET TEST :  [ 0.  0.  1. ...,  0.  0.  1.]
Confusion Matrix:
[[2145  288]
 [ 291  532]]

Accuracy:
82.2174447174 %

Precission:
0.648780487805
```

```
Recall:
0.646415552855


fold ke  6
PREDICTION:  [ 0.  1.  0. ...,  0.  0.  1.]
TARGET TEST :  [ 0.  1.  1. ...,  0.  0.  0.]
Confusion Matrix:
[[2185  297]
 [ 270  504]]

Accuracy:
82.585995086 %

Precission:
0.629213483146

Recall:
0.651162790698


fold ke  7
PREDICTION:  [ 0.  0.  0. ...,  1.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  1.  0.  0.]
Confusion Matrix:
[[2210  306]
 [ 271  469]]

Accuracy:
82.2788697789 %

Precission:
0.605161290323

Recall:
0.633783783784


fold ke  8
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  1.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2164  296]
 [ 306  490]]

Accuracy:
81.5110565111 %
```

```
Precission:
0.623409669211

Recall:
0.615577889447


fold ke  9
PREDICTION:  [ 0.  1.  0. ...,  0.  0.  1.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2140  280]
 [ 317  519]]

Accuracy:
81.6646191646 %

Precission:
0.649561952441

Recall:
0.620813397129


fold ke  10
PREDICTION:  [ 0.  1.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  1.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2143  309]
 [ 319  485]]

Accuracy:
80.7125307125 %

Precission:
0.610831234257

Recall:
0.603233830846


Rata-rata accuracy:  81.7665249823 %
```

In [5]:
```python
#Training dengan MLP , kFold 10 fold , metrics, confusion matrix
import numpy as np
```

```
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
import pandas as pd
from sklearn.datasets import load_svmlight_files
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

#PEMBACAAN DATASET CENSUS
cen = pd.read_csv('DatasetEskperimen/CensusIncome/CencusIncome.data.txt', sep=",", head

census_data = cen[["age", "workclass", "fnlwgt","education","education-num","marital-s
census_target = cen["50K"]


#changing target into float 0 and 1
new = []

for index, item in enumerate(census_target):
    if (item == " <=50K"):
        new.append(0.0)
    else:
        if(item == " >50K"):
            new.append(1.0)
        else:
            new.append(2.0)

new = np.array(new)

new_data = pd.get_dummies(census_data)
new_data = new_data.values


split_number = 10

#folding
kf = KFold(n_splits=split_number,shuffle= True)
jumlah = 0
nomorFold = 1
for train_index,test_index in kf.split(new_data):
    data_train,data_test = new_data[train_index],new_data[test_index]
    target_train,target_test = new[train_index], new[test_index]

    #learning dataset
    clf = MLPClassifier(activation='logistic',max_iter = 1000)
```

```
        clf.fit(data_train,target_train)
        print("fold ke: ",nomorFold)
        #predicting learning data
        prediction = clf.predict(data_test)
        print('PREDICTION: ',prediction)
        print('TARGET TEST : ',target_test)

        #generating confusion matrix
        #conf_matrix =
        conf = confusion_matrix(target_test,prediction)
        print('Confusion Matrix:')
        print(conf)

        #accuracy
        print('\nAccuracy:')
        acc = accuracy_score(target_test,prediction)
        jumlah+=acc
        print(acc*100,'%')

        #precision
        print('\nPrecission:')
        prec = precision_score(target_test,prediction)
        print(prec)

        #recall
        print('\nRecall:')
        rec = recall_score(target_test,prediction)
        print(rec)
        print('\n')
        nomorFold+=1

    average = jumlah/10;
    print('Rata-rata accuracy: ',average*100,'%\n')
```

```
fold ke:  1
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  1.  0. ...,  0.  0.  1.]
Confusion Matrix:
[[2472    0]
 [ 784    1]]

Accuracy:
75.9287688056 %

Precission:
1.0

Recall:
```

```
0.00127388535032


fold ke:  2
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2511    0]
 [ 743    2]]

Accuracy:
77.1805896806 %

Precission:
1.0

Recall:
0.00268456375839


fold ke:  3
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 1.  0.  0. ...,  0.  0.  0.]
Confusion Matrix:
[[2481    0]
 [ 774    1]]

Accuracy:
76.2285012285 %

Precission:
1.0

Recall:
0.00129032258065


fold ke:  4
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  1.  1.]
Confusion Matrix:
[[2479    0]
 [ 777    0]]

Accuracy:
76.1363636364 %

Precission:
```

```
0.0

Recall:
0.0



/home/dicky/miniconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Unde
  'precision', 'predicted', average, warn_for)


fold ke:  5
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  1.  0. ...,  1.  1.  0.]
Confusion Matrix:
[[2472    0]
 [ 770   14]]

Accuracy:
76.3513513514 %

Precission:
1.0

Recall:
0.0178571428571


fold ke:  6
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  1.  0. ...,  1.  1.  1.]
Confusion Matrix:
[[2468    0]
 [ 788    0]]

Accuracy:
75.7985257985 %

Precission:
0.0

Recall:
0.0


fold ke:  7
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
```

```
TARGET TEST :  [ 0.  0.  1. ...,  0.  0.  0.]
Confusion Matrix:
[[2494    0]
 [ 754    8]]

Accuracy:
76.8427518428 %

Precission:
1.0

Recall:
0.010498687664


fold ke:  8
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  1.  0. ...,  0.  0.  1.]
Confusion Matrix:
[[2487    0]
 [ 764    5]]

Accuracy:
76.5356265356 %

Precission:
1.0

Recall:
0.00650195058518


fold ke:  9
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 1.  1.  1. ...,  1.  0.  0.]
Confusion Matrix:
[[2414    0]
 [ 838    4]]

Accuracy:
74.2628992629 %

Precission:
1.0

Recall:
0.00475059382423
```

```
fold ke:  10
PREDICTION:  [ 0.  0.  0. ...,  0.  0.  0.]
TARGET TEST :  [ 0.  0.  0. ...,  0.  1.  0.]
Confusion Matrix:
[[2440    2]
 [ 791   23]]

Accuracy:
75.644963145 %

Precission:
0.92

Recall:
0.0282555282555


Rata-rata accuracy:  76.0910341287 %
```

```
In [6]: #Training dengan Decision Tree , kFold 10 fold , metrics, confusion matrix
        import numpy as np
        from sklearn import datasets
        from sklearn import tree
        import pandas as pd
        from sklearn.externals import joblib
        from sklearn.datasets import load_svmlight_files
        from sklearn.preprocessing import OneHotEncoder
        from sklearn.model_selection import KFold
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import precision_score
        from sklearn.metrics import recall_score

        #PEMBACAAN DATASET CENSUS UNTUK TRAINING
        cen = pd.read_csv('DatasetEskperimen/CensusIncome/CencusIncome.data.txt', sep=",\s", na

        cen.dropna(inplace=True)

        census_data = cen[["age", "workclass", "fnlwgt","education","education-num","marital-st
        census_target = cen["50K"]


        #changing target into float 0 and 1
        new = []
```

```python
for index, item in enumerate(census_target):
    if (item == "<=50K"):
        new.append(0.0)
    else:
        if(item == ">50K"):
            new.append(1.0)
        else:
            new.append(2.0)

new = np.array(new)

new_data = pd.get_dummies(census_data)
list_census = (list(new_data.columns.values))
print(new_data)
new_data = new_data.values



#learning dataset
clf = tree.DecisionTreeClassifier()
clf.fit(new_data,new)

print(clf)

joblib.dump(clf,'clf.pkl') # menyimpan model ke file eksternal
print('Model Saved!')

huehue = joblib.load('clf.pkl') # membaca model dari file eksternal
print('Model Loaded!')
print(huehue)

print(new_data)
```

```
       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
5       37  284582             14             0             0              40
6       49  160187              5             0             0              16
7       52  209642              9             0             0              45
8       31   45781             14         14084             0              50
9       42  159449             13          5178             0              40
10      37  280464             10             0             0              80
11      30  141297             13             0             0              40
12      23  122272             13             0             0              30
13      32  205019             12             0             0              50
```

15	34	245487	4	0	0	45
16	25	176756	9	0	0	35
17	32	186824	9	0	0	40
18	38	28887	7	0	0	50
19	43	292175	14	0	0	45
20	40	193524	16	0	0	60
21	54	302146	9	0	0	20
22	35	76845	5	0	0	40
23	43	117037	7	0	2042	40
24	59	109015	9	0	0	40
25	56	216851	13	0	0	40
26	19	168294	9	0	0	40
28	39	367260	9	0	0	80
29	49	193366	9	0	0	40
30	23	190709	12	0	0	52
31	20	266015	10	0	0	44
...	...	...	...	...	...	...
32526	32	211349	6	0	0	40
32527	22	203715	10	0	0	40
32528	31	292592	9	0	0	40
32529	29	125976	9	0	0	35
32532	34	204461	16	0	0	60
32533	54	337992	13	0	0	50
32534	37	179137	10	0	0	39
32535	22	325033	8	0	0	35
32536	34	160216	13	0	0	55
32537	30	345898	9	0	0	46
32538	38	139180	13	15020	0	45
32540	45	252208	9	0	0	40
32543	45	119199	12	0	0	48
32544	31	199655	14	0	0	30
32545	39	111499	12	0	0	20
32546	37	198216	12	0	0	40
32547	43	260761	9	0	0	40
32548	65	99359	15	1086	0	60
32549	43	255835	10	0	0	40
32550	43	27242	10	0	0	50
32551	32	34066	6	0	0	40
32552	43	84661	11	0	0	45
32553	32	116138	14	0	0	11
32554	53	321865	14	0	0	40
32555	22	310152	10	0	0	40
32556	27	257302	12	0	0	38
32557	40	154374	9	0	0	40
32558	58	151910	9	0	0	40
32559	22	201490	9	0	0	20
32560	52	287927	9	15024	0	40

	workclass_Federal-gov	workclass_Local-gov	workclass_Private \
0	0	0	0
1	0	0	0
2	0	0	1
3	0	0	1
4	0	0	1
5	0	0	1
6	0	0	1
7	0	0	0
8	0	0	1
9	0	0	1
10	0	0	1
11	0	0	0
12	0	0	1
13	0	0	1
15	0	0	1
16	0	0	0
17	0	0	1
18	0	0	1
19	0	0	0
20	0	0	1
21	0	0	1
22	1	0	0
23	0	0	1
24	0	0	1
25	0	1	0
26	0	0	1
28	0	0	1
29	0	0	1
30	0	1	0
31	0	0	1
...	...	...	...
32526	0	0	1
32527	0	0	1
32528	0	0	1
32529	0	0	1
32532	0	0	1
32533	0	0	1
32534	0	0	1
32535	0	0	1
32536	0	0	1
32537	0	0	1
32538	0	0	1
32540	0	0	0
32543	0	1	0
32544	0	0	1
32545	0	1	0
32546	0	0	1

32547	0	0	1
32548	0	0	0
32549	0	0	0
32550	0	0	0
32551	0	0	1
32552	0	0	1
32553	0	0	1
32554	0	0	1
32555	0	0	1
32556	0	0	1
32557	0	0	1
32558	0	0	1
32559	0	0	1
32560	0	0	0

	workclass_Self-emp-inc	...	\
0	0	...	
1	0	...	
2	0	...	
3	0	...	
4	0	...	
5	0	...	
6	0	...	
7	0	...	
8	0	...	
9	0	...	
10	0	...	
11	0	...	
12	0	...	
13	0	...	
15	0	...	
16	0	...	
17	0	...	
18	0	...	
19	0	...	
20	0	...	
21	0	...	
22	0	...	
23	0	...	
24	0	...	
25	0	...	
26	0	...	
28	0	...	
29	0	...	
30	0	...	
31	0	...	
...	...	...	
32526	0	...	

```
32527                         0            ...
32528                         0            ...
32529                         0            ...
32532                         0            ...
32533                         0            ...
32534                         0            ...
32535                         0            ...
32536                         0            ...
32537                         0            ...
32538                         0            ...
32540                         0            ...
32543                         0            ...
32544                         0            ...
32545                         0            ...
32546                         0            ...
32547                         0            ...
32548                         0            ...
32549                         0            ...
32550                         0            ...
32551                         0            ...
32552                         0            ...
32553                         0            ...
32554                         0            ...
32555                         0            ...
32556                         0            ...
32557                         0            ...
32558                         0            ...
32559                         0            ...
32560                         1            ...

       native-country_Portugal  native-country_Puerto-Rico  \
0                            0                           0
1                            0                           0
2                            0                           0
3                            0                           0
4                            0                           0
5                            0                           0
6                            0                           0
7                            0                           0
8                            0                           0
9                            0                           0
10                           0                           0
11                           0                           0
12                           0                           0
13                           0                           0
15                           0                           0
16                           0                           0
17                           0                           0
```

```
18                          0                     0
19                          0                     0
20                          0                     0
21                          0                     0
22                          0                     0
23                          0                     0
24                          0                     0
25                          0                     0
26                          0                     0
28                          0                     0
29                          0                     0
30                          0                     0
31                          0                     0
...                       ...                   ...
32526                       0                     0
32527                       0                     0
32528                       0                     0
32529                       0                     0
32532                       0                     0
32533                       0                     0
32534                       0                     0
32535                       0                     0
32536                       0                     0
32537                       0                     0
32538                       0                     0
32540                       0                     0
32543                       0                     0
32544                       0                     0
32545                       0                     0
32546                       0                     0
32547                       0                     0
32548                       0                     0
32549                       0                     0
32550                       0                     0
32551                       0                     0
32552                       0                     0
32553                       0                     0
32554                       0                     0
32555                       0                     0
32556                       0                     0
32557                       0                     0
32558                       0                     0
32559                       0                     0
32560                       0                     0

       native-country_Scotland  native-country_South  native-country_Taiwan  \
0                            0                     0                      0
1                            0                     0                      0
```

2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0
26	0	0	0
28	0	0	0
29	0	0	0
30	0	0	0
31	0	0	0
...	...	...	...
32526	0	0	0
32527	0	0	0
32528	0	0	0
32529	0	0	0
32532	0	0	0
32533	0	0	0
32534	0	0	0
32535	0	0	0
32536	0	0	0
32537	0	0	0
32538	0	0	0
32540	0	0	0
32543	0	0	0
32544	0	0	0
32545	0	0	0
32546	0	0	0
32547	0	0	0
32548	0	0	0
32549	0	0	0

32550	0	0	0
32551	0	0	0
32552	0	0	0
32553	0	0	1
32554	0	0	0
32555	0	0	0
32556	0	0	0
32557	0	0	0
32558	0	0	0
32559	0	0	0
32560	0	0	0

	native-country_Thailand	native-country_Trinadad&Tobago \
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
28	0	0
29	0	0
30	0	0
31	0	0
...	...	...
32526	0	0
32527	0	0
32528	0	0
32529	0	0

```
32532                                        0                                0
32533                                        0                                0
32534                                        0                                0
32535                                        0                                0
32536                                        0                                0
32537                                        0                                0
32538                                        0                                0
32540                                        0                                0
32543                                        0                                0
32544                                        0                                0
32545                                        0                                0
32546                                        0                                0
32547                                        0                                0
32548                                        0                                0
32549                                        0                                0
32550                                        0                                0
32551                                        0                                0
32552                                        0                                0
32553                                        0                                0
32554                                        0                                0
32555                                        0                                0
32556                                        0                                0
32557                                        0                                0
32558                                        0                                0
32559                                        0                                0
32560                                        0                                0

       native-country_United-States  native-country_Vietnam  \
0                                 1                       0
1                                 1                       0
2                                 1                       0
3                                 1                       0
4                                 0                       0
5                                 1                       0
6                                 0                       0
7                                 1                       0
8                                 1                       0
9                                 1                       0
10                                1                       0
11                                0                       0
12                                1                       0
13                                1                       0
15                                0                       0
16                                1                       0
17                                1                       0
18                                1                       0
19                                1                       0
20                                1                       0
```

21	1	0
22	1	0
23	1	0
24	1	0
25	1	0
26	1	0
28	1	0
29	1	0
30	1	0
31	1	0
...	...	...
32526	1	0
32527	1	0
32528	1	0
32529	1	0
32532	1	0
32533	0	0
32534	1	0
32535	1	0
32536	1	0
32537	1	0
32538	1	0
32540	1	0
32543	1	0
32544	1	0
32545	1	0
32546	1	0
32547	0	0
32548	1	0
32549	1	0
32550	1	0
32551	1	0
32552	1	0
32553	0	0
32554	1	0
32555	1	0
32556	1	0
32557	1	0
32558	1	0
32559	1	0
32560	1	0

	native-country_Yugoslavia
0	0
1	0
2	0
3	0
4	0

5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
28	0
29	0
30	0
31	0
...	...
32526	0
32527	0
32528	0
32529	0
32532	0
32533	0
32534	0
32535	0
32536	0
32537	0
32538	0
32540	0
32543	0
32544	0
32545	0
32546	0
32547	0
32548	0
32549	0
32550	0
32551	0
32552	0

```
32553                          0
32554                          0
32555                          0
32556                          0
32557                          0
32558                          0
32559                          0
32560                          0

[30162 rows x 104 columns]
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')
Model Saved!
Model Loaded!
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')
[[    39  77516     13 ...,      1      0      0]
 [    50  83311     13 ...,      1      0      0]
 [    38 215646      9 ...,      1      0      0]
 ...,
 [    58 151910      9 ...,      1      0      0]
 [    22 201490      9 ...,      1      0      0]
 [    52 287927      9 ...,      1      0      0]]
```

In [7]:
```
#Training dengan Decision Tree , kFold 10 fold , metrics, confusion matrix
import numpy as np
from sklearn import datasets
from sklearn import tree
import pandas as pd
from sklearn.externals import joblib
from sklearn.datasets import load_svmlight_files
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

#PEMBACAAN DATASET CENSUS UNTUK TESTING
```

```
cen = pd.read_csv('DatasetEskperimen/CensusIncome/CencusIncome.test.txt', sep=",\s", na

cen.dropna(inplace=True)

census_data = cen[["age", "workclass", "fnlwgt","education","education-num","marital-s
census_target = cen["50K"]
#print(census_target)

#changing target into float 0 and 1
new = []

for index, item in enumerate(census_target):
    if (item == "<=50K."):
        new.append(0.0)
    else:
        if(item == ">50K."):
            new.append(1.0)
        else:
            new.append(2.0)

new = np.array(new)

new_data = pd.get_dummies(census_data)

print(new_data)
list_target = (list(new_data.columns.values))

empty_list = (list(set(list_census) - set(list_target)))

while (len(empty_list) > 0):
    new_data[empty_list.pop()] = 0

new_data = new_data.values

clf = joblib.load('clf.pkl') # membaca model dari file eksternal
print('Model Loaded!')



#predicting learning data
prediction = clf.predict(new_data)
print('PREDICTION: ',prediction)
print('TARGET TEST : ',new)

#lihat akurasi
print('\nAccuracy:')
acc = accuracy_score(new,prediction)
print(acc*100,'%')
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	\
0	25	226802	7	0	0	40	
1	38	89814	9	0	0	50	
2	28	336951	12	0	0	40	
3	44	160323	10	7688	0	40	
5	34	198693	6	0	0	30	
7	63	104626	15	3103	0	32	
8	24	369667	10	0	0	40	
9	55	104996	4	0	0	10	
10	65	184454	9	6418	0	40	
11	36	212465	13	0	0	40	
12	26	82091	9	0	0	39	
14	48	279724	9	3103	0	48	
15	43	346189	14	0	0	50	
16	20	444554	10	0	0	25	
17	43	128354	9	0	0	30	
18	37	60548	9	0	0	20	
20	34	107914	13	0	0	47	
21	34	238588	10	0	0	35	
23	25	220931	13	0	0	43	
24	25	205947	13	0	0	40	
25	45	432824	9	7298	0	90	
26	22	236427	9	0	0	20	
27	23	134446	9	0	0	54	
28	54	99516	9	0	0	35	
29	32	109282	10	0	0	60	
30	46	106444	10	7688	0	38	
31	56	186651	7	0	0	50	
32	24	188274	13	0	0	50	
33	23	258120	10	0	0	40	
34	26	43311	9	0	0	40	
...	...	...	...	...	...	...	
16248	25	242136	9	0	0	40	
16249	31	112115	9	0	0	40	
16250	49	77132	9	0	0	40	
16252	60	117909	11	7688	0	40	
16253	39	229647	13	0	1669	40	
16254	38	149347	14	0	0	50	
16255	43	23157	14	0	1902	50	
16256	23	93977	9	0	0	40	
16257	73	159691	10	0	0	40	
16258	35	176967	10	0	0	40	
16259	66	344436	9	0	0	8	
16260	27	430340	10	0	0	45	
16261	40	202168	15	15024	0	55	
16262	51	82720	9	0	0	40	
16263	22	269623	10	0	0	40	
16264	64	136405	9	0	0	32	

16266	55	224655	9	0	0	32
16267	38	247547	11	0	0	40
16268	58	292710	12	0	0	36
16269	32	173449	9	0	0	40
16270	48	285570	9	0	0	40
16271	61	89686	9	0	0	48
16272	31	440129	9	0	0	40
16273	25	350977	9	0	0	40
16274	48	349230	14	0	0	40
16275	33	245211	13	0	0	40
16276	39	215419	13	0	0	36
16278	38	374983	13	0	0	50
16279	44	83891	13	5455	0	40
16280	35	182148	13	0	0	60

	workclass_Federal-gov	workclass_Local-gov	workclass_Private	\
0	0	0	1	
1	0	0	1	
2	0	1	0	
3	0	0	1	
5	0	0	1	
7	0	0	0	
8	0	0	1	
9	0	0	1	
10	0	0	1	
11	1	0	0	
12	0	0	1	
14	0	0	1	
15	0	0	1	
16	0	0	0	
17	0	0	1	
18	0	0	1	
20	0	0	1	
21	0	0	1	
23	0	0	1	
24	0	0	1	
25	0	0	0	
26	0	0	1	
27	0	0	1	
28	0	0	1	
29	0	0	0	
30	0	0	0	
31	0	0	0	
32	0	0	0	
33	0	1	0	
34	0	0	1	
...	...	...	...	
16248	0	0	1	

16249	0	0	1
16250	0	0	0
16252	0	0	1
16253	0	0	1
16254	0	0	1
16255	0	1	0
16256	0	0	1
16257	0	0	0
16258	0	0	1
16259	0	0	1
16260	0	0	1
16261	0	0	1
16262	0	0	1
16263	0	0	1
16264	0	0	0
16266	0	0	1
16267	0	0	1
16268	0	0	1
16269	0	0	1
16270	0	0	1
16271	0	0	1
16272	0	0	1
16273	0	0	1
16274	0	1	0
16275	0	0	1
16276	0	0	1
16278	0	0	1
16279	0	0	1
16280	0	0	0

	workclass_Self-emp-inc	...	\
0	0	...	
1	0	...	
2	0	...	
3	0	...	
5	0	...	
7	0	...	
8	0	...	
9	0	...	
10	0	...	
11	0	...	
12	0	...	
14	0	...	
15	0	...	
16	0	...	
17	0	...	
18	0	...	
20	0	...	

```
21                         0           ...
23                         0           ...
24                         0           ...
25                         0           ...
26                         0           ...
27                         0           ...
28                         0           ...
29                         0           ...
30                         0           ...
31                         0           ...
32                         0           ...
33                         0           ...
34                         0           ...
...                      ...           ...
16248                      0           ...
16249                      0           ...
16250                      1           ...
16252                      0           ...
16253                      0           ...
16254                      0           ...
16255                      0           ...
16256                      0           ...
16257                      1           ...
16258                      0           ...
16259                      0           ...
16260                      0           ...
16261                      0           ...
16262                      0           ...
16263                      0           ...
16264                      0           ...
16266                      0           ...
16267                      0           ...
16268                      0           ...
16269                      0           ...
16270                      0           ...
16271                      0           ...
16272                      0           ...
16273                      0           ...
16274                      0           ...
16275                      0           ...
16276                      0           ...
16278                      0           ...
16279                      0           ...
16280                      1           ...

       native-country_Portugal  native-country_Puerto-Rico  \
0                            0                           0
1                            0                           0
```

2	0	0
3	0	0
5	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
20	0	0
21	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
30	0	0
31	0	0
32	0	0
33	0	0
34	0	0
...	...	...
16248	0	0
16249	0	0
16250	0	0
16252	0	0
16253	0	0
16254	0	0
16255	0	0
16256	0	0
16257	0	0
16258	0	0
16259	0	0
16260	0	0
16261	0	0
16262	0	0
16263	0	0
16264	0	0
16266	0	0
16267	0	0
16268	0	0

16269	0	0
16270	0	0
16271	0	0
16272	0	0
16273	0	0
16274	0	0
16275	0	0
16276	0	0
16278	0	0
16279	0	0
16280	0	0

	native-country_Scotland	native-country_South	native-country_Taiwan	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
5	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
14	0	0	0	
15	0	0	0	
16	0	0	0	
17	0	0	0	
18	0	0	0	
20	0	0	0	
21	0	0	0	
23	0	0	0	
24	0	0	0	
25	0	0	0	
26	0	0	0	
27	0	0	0	
28	0	0	0	
29	0	0	0	
30	0	0	0	
31	0	0	0	
32	0	0	0	
33	0	0	0	
34	0	0	0	
...	...	...	...	
16248	0	0	0	
16249	0	0	0	
16250	0	0	0	
16252	0	0	0	

16253	0	0	0
16254	0	0	0
16255	0	0	0
16256	0	0	0
16257	0	0	0
16258	0	0	0
16259	0	0	0
16260	0	0	0
16261	0	0	0
16262	0	0	0
16263	0	0	0
16264	0	0	0
16266	0	0	0
16267	0	0	0
16268	0	0	0
16269	0	0	0
16270	0	0	0
16271	0	0	0
16272	0	0	0
16273	0	0	0
16274	0	0	0
16275	0	0	0
16276	0	0	0
16278	0	0	0
16279	0	0	0
16280	0	0	0

	native-country_Thailand	native-country_Trinadad&Tobago	\
0	0	0	
1	0	0	
2	0	0	
3	0	0	
5	0	0	
7	0	0	
8	0	0	
9	0	0	
10	0	0	
11	0	0	
12	0	0	
14	0	0	
15	0	0	
16	0	0	
17	0	0	
18	0	0	
20	0	0	
21	0	0	
23	0	0	
24	0	0	

```
25                             0                                  0
26                             0                                  0
27                             0                                  0
28                             0                                  0
29                             0                                  0
30                             0                                  0
31                             0                                  0
32                             0                                  0
33                             0                                  0
34                             0                                  0
...                          ...                                ...
16248                          0                                  0
16249                          0                                  0
16250                          0                                  0
16252                          0                                  0
16253                          0                                  0
16254                          0                                  0
16255                          0                                  0
16256                          0                                  0
16257                          0                                  0
16258                          0                                  0
16259                          0                                  0
16260                          0                                  0
16261                          0                                  0
16262                          0                                  0
16263                          0                                  0
16264                          0                                  0
16266                          0                                  0
16267                          0                                  0
16268                          0                                  0
16269                          0                                  0
16270                          0                                  0
16271                          0                                  0
16272                          0                                  0
16273                          0                                  0
16274                          0                                  0
16275                          0                                  0
16276                          0                                  0
16278                          0                                  0
16279                          0                                  0
16280                          0                                  0

       native-country_United-States  native-country_Vietnam  \
0                                 1                       0
1                                 1                       0
2                                 1                       0
3                                 1                       0
5                                 1                       0
```

7	1	0
8	1	0
9	1	0
10	1	0
11	1	0
12	1	0
14	1	0
15	1	0
16	1	0
17	1	0
18	1	0
20	1	0
21	1	0
23	0	0
24	1	0
25	1	0
26	1	0
27	1	0
28	1	0
29	1	0
30	1	0
31	1	0
32	1	0
33	1	0
34	1	0
...	...	...
16248	1	0
16249	1	0
16250	0	0
16252	1	0
16253	1	0
16254	1	0
16255	1	0
16256	1	0
16257	1	0
16258	1	0
16259	1	0
16260	1	0
16261	1	0
16262	1	0
16263	1	0
16264	1	0
16266	1	0
16267	1	0
16268	1	0
16269	1	0
16270	1	0
16271	1	0

16272	1	0
16273	1	0
16274	1	0
16275	1	0
16276	1	0
16278	1	0
16279	1	0
16280	1	0

	native-country_Yugoslavia
0	0
1	0
2	0
3	0
5	0
7	0
8	0
9	0
10	0
11	0
12	0
14	0
15	0
16	0
17	0
18	0
20	0
21	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	0
34	0
...	...
16248	0
16249	0
16250	0
16252	0
16253	0
16254	0
16255	0

```
16256                          0
16257                          0
16258                          0
16259                          0
16260                          0
16261                          0
16262                          0
16263                          0
16264                          0
16266                          0
16267                          0
16268                          0
16269                          0
16270                          0
16271                          0
16272                          0
16273                          0
16274                          0
16275                          0
16276                          0
16278                          0
16279                          0
16280                          0

[15060 rows x 103 columns]
Model Loaded!
PREDICTION:  [ 0.  0.  1. ...,  1.  0.  0.]
TARGET TEST :  [ 0.  0.  1. ...,  0.  0.  1.]

Accuracy:
78.9375830013 %
```