Tubes AI Final

November 20, 2017

1 Tugas Besar IF3170 Inteligensia Buatan 2017/2018

1.1 Aplikasi Web Prediksi Income Per Tahun Tahap 1 : Eksperimen

Anggota : 1. Dewita Sonya Tarabunga - 13515021 2. Helena Suzane Graciella - 13515032 4. Emilia Razak - 13515056 3. Jehian Norman Saviero - 13515139

1.1.1 Analisis Data

Data training yang diberikan merupakan data dari 32561 hasil sensus yang memberikan data-data berupa age, workclass, final weight, education, education number, marital status, occupation, race, sex, capital gain, capital loss, hours per week, dan native country. Terdapat juga data test yang akan digunakan untuk melakukan test hasil pembelajaran.

Analisis yang kami lakukan terhadap data adalah analisis terhadap missing value pada data. Terdapat 3 kolom pada data yang memiliki missing value, yaitu '?' yaitu workclass, occupation, dan native country. Selain itu, semua kolom lain tidak memiliki missing value. Setelah dilihat dari data, semua orang yang memiliki missing value pada occupation hanya memiliki dua kemungkinan nilai workclass, yaitu Never-worked atau missing value. Jelas bahwa jika workclass adalah Never-worked, maka orang tersebut tidak memiliki pekerjaan sehingga value nya kosong. Lalu, jika dilihat dari data yang memiliki missing value di keduanya, hampir 90% dari orang tersebut memiliki gross income <= 50K, maka kami mengasumsikan bahwa orang tersebut pernah bekerja, namun sekarang tidak bekerja sehingga tidak masuk ke kategori manapun dalam workclass. Maka kami mengubah missing value di workclass menjadi Not-worked dan missing value di occupation menjadi None.

Satu kolom lagi yang memiliki missing value adalah native country, namun native country merupakan data categorical sehingga tidak masuk akal jika diganti dengan data mean atau median, begitu pula data modus juga kurang cocok untuk menggantikan missing value. Maka kami menganggap missing value sebagai negara lain di luar daftar, atau Other.

Setelah analisis, kami akan melakukan eksperimen dengan akurasi menjadi ukuran kinerja. Kami akan memilih algoritma dengan akurasi terbaik untuk melakukan full traning dan menghasilkan model untuk kemudian digunakan dalam memrediksi data

1.1.2 Eksperimen : 10-Cross Fold Validation

Import library yang dibutuhkan

```
In [2]: import pandas as pd
        import numpy as np
```

```
from sklearn import preprocessing
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.externals import joblib
```

Membaca data training dari file eksternal

```
In [3]: data = pd.read_csv('CencusIncome.data.txt')
        print(data.head())
```

```
   age   workclass  fnlwgt        edu  edunum              marital occupation  \
0   42  Not-worked   51795    HS-grad       9             Divorced       None
1   60  Not-worked  251572    HS-grad       9              Widowed       None
2   28  Not-worked  201844    HS-grad       9            Separated       None
3   28  Not-worked  196630  Assoc-voc      11            Separated       None
4   67  Not-worked  150264  Doctorate      16  Married-civ-spouse       None

    relationship   race     sex  gain  loss  hours         native  gross
0      Unmarried  Black  Female     0     0     32  United-States  <=50K
1  Not-in-family  White    Male     0     0     35         Poland  <=50K
2      Unmarried  White  Female     0     0     40         Mexico  <=50K
3      Unmarried  White  Female     0     0     40         Mexico  <=50K
4        Husband  White    Male     0     0     20         Canada   >50K
```

Melakukan encoding pada data categorical

```
In [4]: # Digunakan OneHotEncoder pada data categorical selain gross income
        for column in data.columns.values:
            if (data.dtypes[column] == object and column != 'gross'):
                enc = pd.get_dummies(data[column])
                data = data.drop(column, axis=1)
                data = data.join(enc)
        # Digunakan LabelEncoder pada data gross karena kolom tersebut merupakan target
        le = preprocessing.LabelEncoder()
        le.fit(['<=50K', '>50K'])
        enc = le.transform(data['gross'])
        data['gross'] = enc
        print(data.head())
```

```
   age  fnlwgt  edunum  gain  loss  hours  gross  Federal-gov  Local-gov  \
0   42   51795       9     0     0     32      0            0          0
```

```
1   60  251572       9     0     0     35      0            0          0
2   28  201844       9     0     0     40      0            0          0
3   28  196630      11     0     0     40      0            0          0
4   67  150264      16     0     0     20      1            0          0

   Never-worked     ...      Portugal  Puerto-Rico  Scotland  South  Taiwan  \
0             0     ...             0            0         0      0       0
1             0     ...             0            0         0      0       0
2             0     ...             0            0         0      0       0
3             0     ...             0            0         0      0       0
4             0     ...             0            0         0      0       0

   Thailand  Trinadad&Tobago  United-States  Vietnam  Yugoslavia
0         0                0              1        0           0
1         0                0              0        0           0
2         0                0              0        0           0
3         0                0              0        0           0
4         0                0              0        0           0

[5 rows x 109 columns]
```

Melakukan split antara data dan target untuk difit

```
In [5]: datatrain = data.drop('gross', axis=1)
        targettrain = data['gross']
```

Import Confusion Matrix

```
In [6]: from sklearn.metrics import confusion_matrix
```

Melakukan eksperimen dengan Naive Bayes

```
In [7]: kf = KFold(n_splits=10, shuffle=True)
        gnb = GaussianNB()
        for train_index, test_index in kf.split(datatrain):
                X_train, X_test = datatrain.iloc[train_index], datatrain.iloc[test_index]
                Y_train, Y_test = targettrain.iloc[train_index], targettrain.iloc[test_index]
                gnb = gnb.fit(X_train, Y_train)
                pred = gnb.predict(X_test)
                print(accuracy_score(Y_test, pred))
                print(confusion_matrix(Y_test, pred, labels=[0,1]))
```

```
0.805035308566
[[2387  135]
 [ 500  235]]
0.796068796069
[[2331  127]
```

```
 [ 537  261]]
0.792383292383
[[2353  122]
 [ 554  227]]
0.789004914005
[[2327  130]
 [ 557  242]]
0.777641277641
[[2277  148]
 [ 576  255]]
0.785933660934
[[2322  133]
 [ 564  237]]
0.815724815725
[[2406  112]
 [ 488  250]]
0.793304668305
[[2344  116]
 [ 557  239]]
0.808046683047
[[2382  112]
 [ 513  249]]
0.788697788698
[[2327  129]
 [ 559  241]]
```

Melakukan ekperimen dengan Decision Tree Classifier

```
In [8]: kf = KFold(n_splits=10, shuffle=True)
        tclf = tree.DecisionTreeClassifier(criterion="entropy", max_depth=10)
        for train_index, test_index in kf.split(datatrain):
                X_train, X_test = datatrain.iloc[train_index], datatrain.iloc[test_index]
                Y_train, Y_test = targettrain.iloc[train_index], targettrain.iloc[test_index]
                tclf = tclf.fit(X_train, Y_train)
                pred = tclf.predict(X_test)
                print(accuracy_score(Y_test, pred))
                print(confusion_matrix(Y_test, pred, labels=[0,1]))
```

```
0.849247774025
[[2357  138]
 [ 353  409]]
0.865171990172
[[2336  128]
 [ 311  481]]
0.851658476658
[[2331  154]
 [ 329  442]]
```

```
0.857493857494
[[2329  127]
 [ 337  463]]
0.865786240786
[[2343  105]
 [ 332  476]]
0.855343980344
[[2355  132]
 [ 339  430]]
0.855343980344
[[2346  134]
 [ 337  439]]
0.861486486486
[[2362  121]
 [ 330  443]]
0.858415233415
[[2312  124]
 [ 337  483]]
0.847972972973
[[2276  210]
 [ 285  485]]
```

Melakukan ekperimen dengan Multilayer Perceptron

```
In [9]: kf = KFold(n_splits=10, shuffle=True)
        nclf = MLPClassifier(alpha=1e-5, hidden_layer_sizes=(20, 20, 20), random_state=1, max_it
        for train_index, test_index in kf.split(datatrain):
                X_train, X_test = datatrain.iloc[train_index], datatrain.iloc[test_index]
                Y_train, Y_test = targettrain.iloc[train_index], targettrain.iloc[test_index]
                nclf = nclf.fit(X_train, Y_train)
                pred = nclf.predict(X_test)
                print(accuracy_score(Y_test, pred))
                print(confusion_matrix(Y_test, pred, labels=[0,1]))
```

```
0.795517347252
[[2376   96]
 [ 570  215]]
0.785626535627
[[2418   51]
 [ 647  140]]
0.791154791155
[[2328  135]
 [ 545  248]]
0.681511056511
[[1660  816]
 [ 221  559]]
0.789312039312
```

```
[[2431   37]
 [ 649  139]]
0.803132678133
[[2468   34]
 [ 607  147]]
0.764434889435
[[2453    7]
 [ 760   36]]
0.799754299754
[[2444   37]
 [ 615  160]]
0.779176904177
[[2447   18]
 [ 701   90]]
0.773648648649
[[2454   10]
 [ 727   65]]
```

```
In [10]: kf = KFold(n_splits=10, shuffle=True)
         knn = KNeighborsClassifier()
         for train_index, test_index in kf.split(datatrain):
                 X_train, X_test = datatrain.iloc[train_index], datatrain.iloc[test_index]
                 Y_train, Y_test = targettrain.iloc[train_index], targettrain.iloc[test_index]
                 knn = knn.fit(X_train, Y_train)
                 pred = knn.predict(X_test)
                 print(accuracy_score(Y_test, pred))
                 print(confusion_matrix(Y_test, pred, labels=[0,1]))
```

```
0.773411114523
[[2286  228]
 [ 510  233]]
0.772113022113
[[2245  220]
 [ 522  269]]
0.778255528256
[[2271  201]
 [ 521  263]]
0.766584766585
[[2241  224]
 [ 536  255]]
0.779791154791
[[2263  173]
 [ 544  276]]
0.769041769042
[[2251  205]
 [ 547  253]]
0.777948402948
```

```
[[2292  215]
 [ 508  241]]
0.789619164619
[[2310  193]
 [ 492  261]]
0.781941031941
[[2281  187]
 [ 523  265]]
0.769656019656
[[2246  188]
 [ 562  260]]
```

1.1.3 Full Training

Kami menggunakan algoritma DecisionTreeClassifier karena algoritma tersebut memiliki akurasi yang paling tinggi.

Training

```
In [11]: tclf = tree.DecisionTreeClassifier(criterion="entropy", max_depth=10)
         tclf = tclf.fit(datatrain, targettrain)
```

Simpan model ke file eksternal

```
In [12]: joblib.dump(tclf, 'model.pkl')

Out[12]: ['model.pkl']
```

1.1.4 Evaluasi Model

Load model dari file eksternal

```
In [13]: model = joblib.load('model.pkl')
```

Membaca data test dari file eksternal

```
In [14]: test = pd.read_csv('CencusIncome.test.txt')
         print(test.head())

   age   workclass  fnlwgt           edu  edunum             marital  \
0   25     Private  226802          11th       7       Never-married
1   38     Private   89814       HS-grad       9  Married-civ-spouse
2   28   Local-gov  336951    Assoc-acdm      12  Married-civ-spouse
3   44     Private  160323  Some-college      10  Married-civ-spouse
4   58  Not-worked  299831       HS-grad       9  Married-civ-spouse

          occupation relationship   race   sex  gain  loss  hours  \
0  Machine-op-inspct    Own-child  Black  Male     0     0     40
```

```
1    Farming-fishing      Husband  White  Male     0     0     50
2    Protective-serv      Husband  White  Male     0     0     40
3  Machine-op-inspct      Husband  Black  Male  7688     0     40
4               None      Husband  White  Male     0     0     35

          native  gross
0  United-States  <=50K
1  United-States  <=50K
2  United-States   >50K
3  United-States   >50K
4  United-States  <=50K
```

Melakukan encoding pada data categorical

```
In [15]: # Digunakan OneHotEncoder pada data categorical selain gross income
         for column in test.columns.values:
             if (test.dtypes[column] == object and column != 'gross'):
                 enc = pd.get_dummies(test[column])
                 test = test.drop(column, axis=1)
                 test = test.join(enc)
         # Digunakan LabelEncoder pada data gross karena kolom tersebut merupakan target
         le = preprocessing.LabelEncoder()
         le.fit(['<=50K', '>50K'])
         enc = le.transform(test['gross'])
         test['gross'] = enc
         print(test.head())
```

```
   age  fnlwgt  edunum  gain  loss  hours  gross  Federal-gov  Local-gov  \
0   25  226802       7     0     0     40      0            0          0
1   38   89814       9     0     0     50      0            0          0
2   28  336951      12     0     0     40      1            0          1
3   44  160323      10  7688     0     40      1            0          0
4   58  299831       9     0     0     35      0            0          0

   Never-worked     ...      Portugal  Puerto-Rico  Scotland  South  Taiwan  \
0             0     ...             0            0         0      0       0
1             0     ...             0            0         0      0       0
2             0     ...             0            0         0      0       0
3             0     ...             0            0         0      0       0
4             0     ...             0            0         0      0       0

   Thailand  Trinadad&Tobago  United-States  Vietnam  Yugoslavia
0         0                0              1        0           0
1         0                0              1        0           0
2         0                0              1        0           0
3         0                0              1        0           0
4         0                0              1        0           0
```

```
[5 rows x 108 columns]
```

Menambah kolom yang kurang pada data test Terdapat satu kategori pada native country yang tidak muncul pada data test, yaitu Holand-Netherlands. Akibatnya saat dilakukan OneHotEncoding, kolom tersebut tidak muncul, padahal kolom dari data training dan test harus sama. Maka solusinya adalah menambah kolom Holand-Netherlands yang berisi 0 pada setiap row.

```
In [16]: test['Holand-Netherlands'] = 0
         test = test[data.columns.tolist()]
```

Melakukan split antara data dan target untuk diprediksi

```
In [17]: datatest = test.drop('gross', axis=1)
         targettest = test['gross']
```

Melakukan prediksi

```
In [18]: pred = model.predict(datatest)
         print(pred)
         print(accuracy_score(targettest, pred))
         print(confusion_matrix(targettest, pred, labels=[0,1]))
```

```
[0 0 0 ..., 0 0 0]
0.860205147104
[[11821   614]
 [ 1662  2184]]
```