

Tubes2A_13515062

November 28, 2017

Tugas Besar 2 Intelijensi Buatan

Prediksi Income per Tahun

Website: <http://predicio.herokuapp.com/>

Anggota Kelompok: - Devin Alvaro / 13515062 - Stevanno Hero Leadervand / 13515082 - Rizki Ihza / 13515104 - Gianfranco Fertino Hwandiano / 13515118

1 Libraries

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

from sklearn import preprocessing, neighbors, tree
from sklearn.externals import joblib
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier

%matplotlib inline
```

2 Membaca dataset

```
In [2]: train_df = pd.read_csv("data/CencusIncome.data.txt", header = None)
temp = pd.read_csv("data/CencusIncome.data.txt", header = None)
# name columns
train_df = train_df.rename(columns={0: 'age', 1: 'workclass', 2: 'fnlwgt', 3: 'education', 4: 'sex', 5: 'marital', 6: 'occupation', 7: 'relationship', 8: 'race', 9: 'ethnicity', 10: 'native', 11: 'foreignborn', 12: 'nativity', 13: 'region', 14: 'education-num', 15: 'experience', 16: 'experience-since', 17: 'unemployed', 18: 'employed', 19: 'retired', 20: 'disabled', 21: 'other', 22: 'income'})
temp = temp.rename(columns={0: 'age', 1: 'workclass', 2: 'fnlwgt', 3: 'education', 4: 'sex', 5: 'marital', 6: 'occupation', 7: 'relationship', 8: 'race', 9: 'ethnicity', 10: 'native', 11: 'foreignborn', 12: 'nativity', 13: 'education-num', 14: 'experience', 15: 'experience-since', 16: 'unemployed', 17: 'employed', 18: 'retired', 19: 'disabled', 20: 'other', 21: 'income'})
```

3 Preprocessing

3.1 Feature Selection

```
In [3]: from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

le = preprocessing.LabelEncoder()
```

```

for column in temp:
    le.fit(temp[column])
    temp[column] = le.transform(temp[column])

y = np.array(temp['label'])
x = np.array(temp.drop(['label'], 1))

#feature extraction
model = LogisticRegression()
rfe = RFE(model, 1)
fit = rfe.fit(x, y)
print("Feature Ranking: ")
print(fit.ranking_)

```

Feature Ranking:

```
[ 6 10 14 11  2  3 12  4  5  1  9  8  7 13]
```

Feature selection merupakan metode untuk memilih subset atribut-atribut/features dari dataset yang dirasa penting saja. Manfaatnya diantara lain yaitu meningkatkan akurasi model dan mempercepat proses training karena data menjadi lebih sedikit.

Metode feature selection yang kelompok kami gunakan yaitu Recursive Feature Elimination (RFE). Metode ini mengurutkan atribut-atribut (ranking) dari urutan 1 (paling penting) hingga seterusnya (semakin tidak penting).

Dari informasi di atas, dapat disimpulkan bahwa fnlwgt (atribut nomor 3 dari kiri) merupakan atribut yang paling tidak penting (urutan terakhir yaitu 14) jika dibandingkan dengan atribut-atribut lainnya. Berdasarkan hal tersebut, kami memilih untuk tidak mengikutkan atribut fnlwgt pada training model kami.

```

In [4]: # remove 'fnlwgt' column
        train_df = train_df.drop(['fnlwgt'], axis=1)

```

Missing Value Treatment

Dari dataset, kami melihat bahwa ada sekitar 4000 baris data yang mengandung missing value, ditandai dengan tanda ?.

Kami telah melakukan berbagai eksplorasi tentang bagaimana menghandle hal tersebut.

Kami telah mencoba menghapus data yang bersangkutan dan juga mengganti value ? dengan value modus ataupun rata2 dari atribut yang bersangkutan.

Namun, semua hal tersebut tidak meningkatkan akurasi dari model kami, malah justru menurunkan akurasi. Jadi, kami memutuskan untuk membiarkan value tersebut.

3.2 Encoding data dengan One-Hot Encoding

```

In [5]: le = preprocessing.LabelEncoder()

        le.fit(train_df['label'])
        train_df['label'] = le.transform(train_df['label'])

```

```
train_df = pd.get_dummies(train_df)

y = np.array(train_df['label'])
x = np.array(train_df.drop(['label'], 1))
```

One-Hot Encoding

Pada dataset yang diberikan, terdapat nilai-nilai berupa string (bukan angka), sehingga harus dilakukan pemrosesan (*encoding*) agar nilai-nilai string tersebut menjadi angka yang dapat dikalkulasi oleh algoritma-algoritma *learning* menjadi model.

Untuk *preprocessing* dataset ini, kami memilih *one-hot encoding*, yaitu *preprocessing* yang mentransformasi nilai-nilai pada setiap kolom bernilai string menjadi kolomnya masing-masing. Kemudian, pada tiap kolom baru ini terdapat nilai biner (1 atau 0) yang menandakan suatu data berada dalam kategori bernilai demikian atau bukan.

Sebagai contoh, terdapat kolom Sex dengan nilai Male dan Female. Kemudian, kolom Sex ini ditransformasi menjadi 2 kolom, yaitu kolom Male dan Female. Untuk suatu data bernilai Male, maka nilai di kolom Male adalah 1 dan di kolom Female adalah 0. Berlaku pula kebalikannya untuk data bernilai Female.

Kami memilih *one-hot encoding* karena nilai-nilai berbentuk string pada dataset ini tidak tepat bila diubah menjadi angka. Misalnya saja bila pada kolom Race, nilai Asian menjadi 0, White menjadi 1, dan Black menjadi 2, menjadi tidak benar karena menandakan suatu ras bernilai lebih tinggi atau rendah dari ras lain.

One-hot encoding ini memang memiliki kelemahan yaitu menambah jumlah kolom secara signifikan sehingga meningkatkan kompleksitas algoritma *learning* yang ada, namun masih dapat ditoleransi.

3.3 Eksperimen untuk mendapatkan model terbaik

3.3.1 Naive Bayes

```
In [6]: gnb = GaussianNB()

score = cross_val_score(gnb, x, y, cv=10)

In [7]: for i in range(10):
        print("Fold-" + str(i + 1) + ":", "%0.6f" % score[i])

        print()

        print("Mean: %0.6f" % score.mean())
        print("Accuration: %0.6f (+/- %0.6f)" % (score.mean(), score.std() * 2))
```

```
Fold-1: 0.802272
Fold-2: 0.804054
Fold-3: 0.800061
Fold-4: 0.795762
Fold-5: 0.796376
Fold-6: 0.803440
Fold-7: 0.793305
Fold-8: 0.802826
```

Fold-9: 0.806204
Fold-10: 0.806204

Mean: 0.801050
Accuration: 0.801050 (+/- 0.008562)

3.3.2 Decision Tree

```
In [8]: ID3learn = tree.DecisionTreeClassifier(criterion="entropy")

        score = cross_val_score(ID3learn, x, y, cv=10)

In [9]: for i in range(10):
        print("Fold-" + str(i + 1) + ":", "%0.6f" % score[i])

        print()

        print("Mean: %0.6f" % score.mean())
        print("Accuration: %0.6f (+/- %0.6f)" % (score.mean(), score.std() * 2))
```

Fold-1: 0.817931
Fold-2: 0.825553
Fold-3: 0.835381
Fold-4: 0.812654
Fold-5: 0.828317
Fold-6: 0.826781
Fold-7: 0.821560
Fold-8: 0.825246
Fold-9: 0.834152
Fold-10: 0.820639

Mean: 0.824821
Accuration: 0.824821 (+/- 0.013252)

3.3.3 k-Nearest Neighbors

```
In [10]: n_neighbors = 61

        KNNlearn = neighbors.KNeighborsClassifier(n_neighbors, weights='uniform')

        score = cross_val_score(KNNlearn, x, y, cv=10)

In [11]: for i in range(10):
        print("Fold-" + str(i + 1) + ":", "%0.6f" % score[i])

        print()
```

```

print("Mean: %0.6f" % score.mean())
print("Accuration: %0.6f (+/- %0.6f)" % (score.mean(), score.std() * 2))

Fold-1: 0.842186
Fold-2: 0.847973
Fold-3: 0.852887
Fold-4: 0.841523
Fold-5: 0.838145
Fold-6: 0.853808
Fold-7: 0.843059
Fold-8: 0.851658
Fold-9: 0.851044
Fold-10: 0.854115

Mean: 0.847640
Accuration: 0.847640 (+/- 0.011201)

```

3.3.4 Multilayer Perceptron

```

In [12]: MLPlearn = MLPClassifier(solver='lbfgs',hidden_layer_sizes=(5, 2))

score = cross_val_score(MLPlearn, x, y, cv=10)

In [13]: for i in range(10):
          print("Fold-" + str(i + 1) + ":", "%0.6f" % score[i])

          print()

          print("Mean: %0.6f" % score.mean())
          print("Accuration: %0.2f (+/- %0.6f)" % (score.mean(), score.std() * 2))

Fold-1: 0.809334
Fold-2: 0.240786
Fold-3: 0.757985
Fold-4: 0.797912
Fold-5: 0.800676
Fold-6: 0.759214
Fold-7: 0.820639
Fold-8: 0.817260
Fold-9: 0.830160
Fold-10: 0.759214

Mean: 0.739318
Accuration: 0.74 (+/- 0.336282)

```

Memilih model terbaik

setelah dilakukan percobaan learning dengan beberapa algoritma yaitu:

- Naive Bayes
- Decision Tree Learning
- K-Nearest neighbours
- Multilayer Perceptron

didapat model yang memiliki akurasi tertinggi untuk dataset ini adalah model K-Nearest neighbours. Oleh karena itu dipilih model K-Nearest neighbour untuk digunakan dalam perhitungan selanjutnya

```
In [14]: KNNlearn = neighbors.KNeighborsClassifier(n_neighbors, weights='uniform')
         KNNlearn.fit(x, y)
```

```
Out[14]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                             metric_params=None, n_jobs=1, n_neighbors=61, p=2,
                             weights='uniform')
```

3.4 Menyimpan model

```
In [15]: joblib.dump(KNNlearn, 'model/best.pkl')
         joblib.dump(KNNlearn, '../webapp/model/best.pkl')
```

```
Out[15]: ['../webapp/model/best.pkl']
```

3.5 Loading model

```
In [16]: KNNlearn = joblib.load('model/best.pkl')
```

3.6 Evaluasi dan prediksi dengan model terpilih

3.6.1 Membaca test dataset

```
In [17]: test_df = pd.read_csv("data/CencusIncome.test.txt", header=None, skiprows=1)

         # name columns
         test_df = test_df.rename(columns={0: 'age', 1: 'workclass', 2: 'fnlwgt', 3: 'education'

         # remove 'fnlwgt' column
         test_df = test_df.drop(['fnlwgt'], axis=1)

         test_df.head(10)
```

```
Out[17]:
```

	age	workclass	education	education-num	marital-status \
0	38	Private	HS-grad	9	Married-civ-spouse
1	28	Local-gov	Assoc-acdm	12	Married-civ-spouse
2	44	Private	Some-college	10	Married-civ-spouse
3	18	?	Some-college	10	Never-married
4	34	Private	10th	6	Never-married
5	29	?	HS-grad	9	Never-married
6	63	Self-emp-not-inc	Prof-school	15	Married-civ-spouse

7	24	Private	Some-college	10	Never-married
8	55	Private	7th-8th	4	Married-civ-spouse
9	65	Private	HS-grad	9	Married-civ-spouse

	occupation	relationship	race	sex	capital-gain \
0	Farming-fishing	Husband	White	Male	0
1	Protective-serv	Husband	White	Male	0
2	Machine-op-inspct	Husband	Black	Male	7688
3	?	Own-child	White	Female	0
4	Other-service	Not-in-family	White	Male	0
5	?	Unmarried	Black	Male	0
6	Prof-specialty	Husband	White	Male	3103
7	Other-service	Unmarried	White	Female	0
8	Craft-repair	Husband	White	Male	0
9	Machine-op-inspct	Husband	White	Male	6418

	capital-loss	hours-per-week	native-country	label
0	0	50	United-States	<=50K
1	0	40	United-States	>50K
2	0	40	United-States	>50K
3	0	30	United-States	<=50K
4	0	30	United-States	<=50K
5	0	40	United-States	<=50K
6	0	32	United-States	>50K
7	0	40	United-States	<=50K
8	0	10	United-States	<=50K
9	0	40	United-States	>50K

3.6.2 Preprocessing test dataset

```
In [18]: le = preprocessing.LabelEncoder()

le.fit(test_df['label'])
test_df['label'] = le.transform(test_df['label'])

In [19]: test_df = pd.get_dummies(test_df)

missing_columns = set(train_df.columns) - set(test_df.columns)
for column in missing_columns:
    test_df[column] = 0

y = np.array(test_df['label'])
x = np.array(test_df.drop(['label'], 1))

print(np.shape(y))
print(np.shape(x))

test_df.head(10)
```

```
(16280,)
(16280, 107)
```

```
Out[19]:
```

	age	education-num	capital-gain	capital-loss	hours-per-week	label	\
0	38	9	0	0	50	0	
1	28	12	0	0	40	1	
2	44	10	7688	0	40	1	
3	18	10	0	0	30	0	
4	34	6	0	0	30	0	
5	29	9	0	0	40	0	
6	63	15	3103	0	32	1	
7	24	10	0	0	40	0	
8	55	4	0	0	10	0	
9	65	9	6418	0	40	1	

	workclass_?	workclass_Federal-gov	workclass_Local-gov	\
0	0	0	0	
1	0	0	1	
2	0	0	0	
3	1	0	0	
4	0	0	0	
5	1	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	

	workclass_Never-worked	...	\
0	0	...	
1	0	...	
2	0	...	
3	0	...	
4	0	...	
5	0	...	
6	0	...	
7	0	...	
8	0	...	
9	0	...	

	native-country_Puerto-Rico	native-country_Scotland	native-country_South	\
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

	native-country_Taiwan	native-country_Thailand \
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

	native-country_Trinidad&Tobago	native-country_United-States \
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
5	0	1
6	0	1
7	0	1
8	0	1
9	0	1

	native-country_Vietnam	native-country_Yugoslavia \
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

	native-country_Holand-Netherlands
0	0
1	0
2	0
3	0
4	0
5	0
6	0

7	0
8	0
9	0

[10 rows x 108 columns]

3.6.3 Hasil prediksi

```
In [20]: score = KNNlearn.score(x, y)
         print("Accuracy: ", score * 100 ,"%")

         y_pred = KNNlearn.predict(x)

         print("Confusion Matrix: ")
         print(confusion_matrix(y, y_pred))
```

Accuracy: 85.0982800983 %

Confusion Matrix:

```
[[11667  767]
 [ 1659 2187]]
```