Tugas Besar IF3170

Aplikasi Web Prediksi Income Per Tahun

Nama Kelompok : 49

Anggota :

1. Finiko Kasula Novenda - 13515029
2. M. Dicky Andika Putra - 13515044
3. Vincent Hendryanto Halim - 13515089
4. Mikhael Artur Darmakesuma - 13515099
5. William - 13515144

Hasil Analisis Data

Jenis data

Data yang tersedia dapat diklasifikasikan menjadi 2 jenis, yaitu data kontinu dan data kategorial.

- Kolom yang mengandung data kontinu antara lain: age, fnlwgt,education-num, capital-gain, capital-loss, hours-per-week
- Kolom yang mengandung data kategorial antara lain: workclass,education,marital-status,occupation, relationship, race, sex, native-country

Penanganan data tidak lengkap

Dari data awal, masih ada data yang mengandung informasi yang tidak lengkap, ditandai dengan karakter '?' Data yang hilang ini kami ubah menjadi modus keseluruhan data. Data yang hilang hanya terdapat pada kolom data kategorial, dan data kategorial tidak dapat dihitung mean, sehingga dipilih modus.

Perubahan menjadi data numerik

Classifier yang tersedia pada scikit, baik KNN, ID3, MLP maupun NaiveBayes hanya dapat menangani data bersifat numerik. Untuk itu data-data yang bersifat kategorial perlu diubah. Data kategorial ini diubah menggunakan fitur get_dummies pada panda. Dengan get_dummies ini, akan terbentuk kolom baru sebesar sparsity suatu kolom kategorial. Misalkan kolom "Sex" memiliki nilai Male atau Female. Setelah diubah, kolom Sex akan berubah menjadi 2 kolom baru yaitu kolom Sex_Male, dan kolom Sex_Female. Jika sebelumnya kolom Sex bernilai "Male", setelah perubahan kolom Sex_Male akan bernilai 1 dan kolom Sex_Female bernilai 0.

Penanganan data tambahan

Kolom fnlwgt, race, dan native-country didrop karena kolom tersebut dinilai tidak representatif untuk kelas hasil sesuai dengan analisis menggunakan histogram pada

Ukuran Kinerja

Kinerja pada tugas kali ini diukur dengan menggunakan fungsi bawaan sklearn yaitu tree_model.score yang menghitung akurasi dari model terhadap data tes yang diberikan. Akurasi dihitung berdasarkan data yang berhasil diprediksi dengan benar dibanding total semua data. *Accuracy* = (Tp + Tn)/(Tp + Fp + Tn + Fn).

In [8]:
```
# Library Import
import pandas as pd
from pandas import DataFrame
import graphviz
from sklearn import preprocessing
import pickle

# Algorithm
from sklearn.naive_bayes import GaussianNB
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
```

1. Preprocessing Data

In [9]:
```
atributeNames = ["age","workclass","fnlwgt","education","education-num","marital-status","occupation","relationship","race","sex","capital-gain","capital-loss","hours-per-week","native-country","target"]
income = pd.read_csv('data/CensusIncome.data.csv', header=None, names = atributeNames, sep = ",\s", engine="python", na_values="?");
```

Mengubah NaN menjadi Most Frequent

In [10]:
```
income = income.apply(lambda x:x.fillna(x.value_counts().index[0]))
```

Drop Column

In [11]:
```
income = income.drop(columns=['fnlwgt', 'race', 'native-country'])
```

Panda get_dummy

In [12]:
```
incomeFinal = pd.get_dummies(income, columns=["workclass", "education","marita
l-status","occupation","relationship","sex"])
x = incomeFinal[[i for i in list(incomeFinal.columns) if i != 'target']].value
s
y = incomeFinal['target'].values
```

2. Pemilihan Algoritma

2.1 KNN

In [13]:
```
nbrs = KNeighborsClassifier(n_neighbors=28)
score = cross_val_score(nbrs, x, y, cv=10)
print("Score : %.3f" % (score.mean()*100))
```

```
Score : 85.351
```

In [14]:
```
y_pred = cross_val_predict(nbrs,x,y,cv=10)
print(confusion_matrix(y,y_pred))
```

```
[[23232  1488]
 [ 3282  4559]]
```

2.2 MLP

In [21]:
```
mlp = MLPClassifier(solver='adam', alpha=1e-3, hidden_layer_sizes=(10,70), ran
dom_state=1)
score = cross_val_score(mlp, x, y, cv=10).mean()
print("Score : %.3f" % (score.mean()*100))
```

```
Score : 83.873
```

In [22]:
```
y_pred = cross_val_predict(mlp, x, y, cv=10)
print(confusion_matrix(y,y_pred))
```

```
[[22796  1924]
 [ 3327  4514]]
```

2.3 Naive Bayes

In [23]:
```
foldNaiveBayes = GaussianNB()
score = cross_val_score(foldNaiveBayes, x, y, cv = 10)
print("Score : %.3f" % (score.mean()*100))
```

```
Score : 81.367
```

In [24]:
```
y_pred = cross_val_predict(foldNaiveBayes, x, y, cv = 10)
print(confusion_matrix(y,y_pred))
```

```
[[20392  4328]
 [ 1739  6102]]
```

2.4 Decision Tree ID3

In [15]:
```
dt = tree.DecisionTreeClassifier(criterion='entropy', max_features=28) #nilai
 28 diambil dari percobaan
score = cross_val_score(dt, x, y, cv = 10)
print("Score : %.3f" % (score.mean()*100))
```

```
Score : 82.203
```

In [16]:
```
y_pred = cross_val_predict(dt, x, y, cv = 10)
print(confusion_matrix(y,y_pred))
```

```
[[21993  2727]
 [ 3119  4722]]
```

3. Full Training

Dipilih algoritma Decision Tree ID3 Karena menghasilkan nilai akurasi yang paling baik

In [25]:
```
#model_out = tree.DecisionTreeClassifier(criterion='entropy', max_features=28)
 # Mungkin pakai max_features?
model_out = KNeighborsClassifier(n_neighbors=28)
model_out = model_out.fit(x,y)

pickle.dump(model_out,open("out.pkl", "wb+"))
```

In [26]:
```
test_data = pd.read_csv('data/CensusIncome.test.csv', header=None, names = atr
ibuteNames, sep = ",\s", engine="python", na_values="?", comment="|");
test_data = test_data.apply(lambda x:x.fillna(x.value_counts().index[0]))
test_data = test_data.drop(columns=['fnlwgt','race', 'native-country'])
test_data = pd.get_dummies(test_data, columns=["workclass","education","marita
l-status","occupation","relationship","sex"])

test_dataframe = pd.DataFrame(columns = incomeFinal.columns)
test_dataframe.append(test_data)

test_dataframe = pd.concat([test_dataframe, test_data], axis=0)
test_dataframe = test_dataframe.fillna(0)

x_test = test_dataframe[[i for i in list(test_dataframe.columns) if i != 'targ
et']].values
y_test = pd.DataFrame(test_dataframe['target'].values)

model = pickle.load(open("out.pkl","rb+"))
```

In [27]:
```
score = model.score(x_test, y_test)
print("Score : %.2f" % (score*100))
```

```
Score : 85.59
```

In [28]:
```
y_pred = model.predict(x_test)
print(confusion_matrix(y_test,y_pred))
```

```
[[11685   751]
 [ 1596  2250]]
```