Data-Science-Project-Finished

September 25, 2019

1 Data Science Project

Anda diminta oleh pimpinan untuk membuat model prediksi klasifikasi atas `income` pada dataset **Adult Income**. Data ini diperoleh dari Machine Learning Repository UCI (https://archive.ics.uci.edu/ml/datasets/Adult).Target variabel (`income`) merupakan data categorical dengan 2 nilai `<=50k` dan `>50k`.

Kemudian, Anda diminta untuk membandingkan model yang paling baik berdasarkan eksperimen yang Anda lakukan terhadap performa model.

Langkah pertama adalah *import library*.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
# import dataset
df_adult = pd.read_csv('adult.csv')
```

[3]:
```
df_adult.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
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
educational-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
gender             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
income             48842 non-null object
```

```
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

Apakah kita langsung dapat menggunakan data ini secara langsung?
Tentu Tidak !
Kali ini, kita akan memperkenalkan tahapan-tahapan yang akan Anda lakukan sebagai data scientist. Pada umumnya, tahapan yang bisa kita lakukan setidaknya terdiri dari beberapa tahapan berikut ini:

1. Data Cleaning atau Preprocessing
2. Explorasi Data
3. Pengolahan Data
4. Evaluasi

Kali ini, kita akan melakukan tahapan 1 dan 2 terlebih dahulu.

1.1 1. Data Cleaning / Data Preprocessing / Data Wrangling

Tahap pertama adalah melakukan pembersihan data, tahapan ini penting karena sebagian besar pekerjaan Data Analis berkutat pada bagian ini dan tahap explorasi data.

Di tahapan ini hal yang pertama kali dilakukan adalah melakukan *peek* data menggunakan fungsi `head`.

[4]:
```
df_adult.head()
```

[4]:

	age	workclass	fnlwgt	education	educational-num	marital-status	\
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	18	?	103497	Some-college	10	Never-married	

	occupation	relationship	race	gender	capital-gain	capital-loss	\
0	Machine-op-inspct	Own-child	Black	Male	0	0	
1	Farming-fishing	Husband	White	Male	0	0	
2	Protective-serv	Husband	White	Male	0	0	
3	Machine-op-inspct	Husband	Black	Male	7688	0	
4	?	Own-child	White	Female	0	0	

	hours-per-week	native-country	income
0	40	United-States	<=50K
1	50	United-States	<=50K
2	40	United-States	>50K
3	40	United-States	>50K
4	30	United-States	<=50K

[5]:
```
df_adult.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
```

```
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
educational-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
gender             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
income             48842 non-null object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

Dari data tersebut di atas, terdapat 2 jenis data yaitu `int64` dan `object`. Tipe data `object` pada umumnya dapat berupa `String` yang juga pada umumnya, data yang bersifat categorical. Dari kolom tersebut di atas, dapat kita ketahui bahwa terdapat 48.842 jenis data.

Hal yang paling penting di tahapan ini adalah: 1. Cek atas data **NULL** Data **NULL** tersebut bisa berbentuk tanda -, spasi, `NA` ataupun tanda lainnya. Strategi atas data **NULL** tersebut bisa kita lakukan *impute* atau pengisian, atau bisa dihilangkan seluruh baris (row) data tersebut. 2. Cek atas data anomali (outlier) Data anomali dapat dianalisis dengan metode *standard deviasi*. Normalnya, data yang bernilai bilangan riil bisa kita cari dan hilangkan agar tidak mempengaruhi performa dari model kita.

Data anomali lainnya, bisa berupa **kesalahan ketik (typo)** ataupun nilai yang sebenarnya sama tetapi dituliskan berbeda.

Oleh karena itu, kita bisa melakukan cek satu per satu untuk kolom berikut ini:

1. `workclass`
2. `education`
3. `marital-status`
4. `occupation`
5. `relatiohship`
6. `race`
7. `gender`
8. `native-country`
9. `income`

[6]:
```
col = ['workclass', 'education', 'marital-status', 'occupation',␣
 ↪'relationship', 'race', 'gender','native-country']
for c in col:
    print(c,':',df_adult[c].unique())
```

```
workclass : ['Private' 'Local-gov' '?' 'Self-emp-not-inc' 'Federal-gov' 'State-
gov'
```

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

Namun, kita hanya akan fokus terhadap 4 kolom saja yaitu `workclass`, `marital-status`, `country`, `occupation`, karena terlihat dari data categorical di atas, ada beberapa anomali salah satunya adalah nilai `?`.

1.1.1 Kolom workclass

[7]:
```
# kolom workclass
df_adult.workclass.value_counts()
```

[7]:
```
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
```

Ada beberapa **anomali** pada value di dalam kolom `workclass` di antaranya, nilai `?` dan nilai `Self-emp-inc` yang sebenarnya sama dengan `Self-emp-not-inc`.

Solusi: 1. Lakukan perubahan pada nilai `?` menjadi `NA`, karena nilai `?` tidak dapat didefinisikan. 2. Lakukan *merge* untuk nilai kategori pada `Self-emp-inc` dan `Self-emp-not-inc`.

Lakukan cek nilai `NA` apakah ada di dalam dataset.

[8]:
```
# cek NA value di kolom workclass
df_adult.workclass.isnull().any()
```

[8]:
```
False
```

Untuk mengubah nilai ? menjadi NA ada **banyak** cara, salah satunya adalah

```
# Menggunakan klausa where
df_adult.workclass = np.where(df_adult.workclass.str.startswith('?'),np.nan,df_adult.workclass)
# menggunakan apply method dan lambda
df_adult.workclass = df_adult.workclass.apply(lambda x: np.nan if x.startswith('?') else x)
# cara sederhana
df_adult.workclass[df_adult.workclass.str.startswith('?')] = np.nan
# cara map
df_adult.workclass = df_adult.workclass.map(lambda v: np.nan if v == '?' else v)
# cara replace
df_adult.workclass = df_adult.workclass.replace(['?'],np.nan)
```

Anda jangan terpaku pada salah satu cara, bahkan Anda bisa membuat cara baru.

[9]:
```
# menggunakan apply method dan lambda
df_adult.workclass = df_adult.workclass.apply(lambda x: np.nan if x.
 ↪startswith('?') else x)

# cek atas perubahan yang sudah kita lakukan
df_adult.workclass.unique()
```

[9]:
```
array(['Private', 'Local-gov', nan, 'Self-emp-not-inc', 'Federal-gov',
       'State-gov', 'Self-emp-inc', 'Without-pay', 'Never-worked'],
      dtype=object)
```

Sekarang, kita memiliki data **NULL** !

[10]:
```
# cek NA value di kolom workclass
df_adult.workclass.isnull().any()
```

[10]:
```
True
```

Untuk melakukan merge atas nilai kategori Self-emp-inc dan Self-emp-not-inc menjadi Self-employed kita menggunakan metode yang sama seperti di atas. Namun, kali ini kita akan menggunakan cara yang sedikit berbeda, yaitu dengan membuat suatu fungsi baru yaitu change_workclass. Di sini, fungsi ini akan di-passing sebagai *argument* dalam method apply. Begitu juga dengan Government.

[11]:
```
l = list(range(10))
```

[12]:
```
def change_selfemployed(x):
    if x =='Self-emp-not-inc':
        return 'Self-employed'
    elif x =='Self-emp-inc':
        return 'Self-employed'
    else:
        return x
```

```
def change_government(x):
    if x =='Local-gov' or x == 'Federal-gov' or x=='State-gov':
        return 'Gov'
    else:
        return x
```

[13]:
```
df_adult.workclass = df_adult.workclass.apply(change_selfemployed)

df_adult.workclass = df_adult.workclass.apply(change_government)

# cek atas perubahan yang sudah kita lakukan
df_adult.workclass.unique()
```

[13]:
```
array(['Private', 'Gov', nan, 'Self-employed', 'Without-pay',
       'Never-worked'], dtype=object)
```

1.1.2 Kolom Marital-status

Kita mau mengubah status pernikahan menjadi 3 saja, `Married`, `Not-married`, dan `Never-married`.

[14]:
```
df_adult['marital-status'].value_counts()
```

[14]:
```
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
```

[15]:
```
def change_married_status(x):
    if x == 'Never-married':
        return x
    elif x == 'Divorced' or x == 'Separated' or x == 'Widowed':
        return 'Not-Married'
    else:
        return 'Married'
```

[16]:
```
df_adult['marital-status'] = df_adult['marital-status'].
 ↪apply(change_married_status)
df_adult['marital-status'].unique()
```

[16]:
```
array(['Never-married', 'Married', 'Not-Married'], dtype=object)
```

1.1.3 Kolom Country

Dari data `native-country` terlihat bahwa kita bisa menginginkan untuk melakukan **grouping** atas negara-negara asal tersebut ke dalam bentuk region yang sama, hal ini bertujuan untuk

memberikan untuk *memberikan tambahan kekuatan* atas fitur `native-country`. Tentunya, hal ini merupakan diskresi dari masing-masing analis.

[17]:
```
df_adult['native-country'].value_counts()
```

[17]:
```
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
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
Yugoslavia                       23
Laos                             23
Scotland                         21
Honduras                         20
Hungary                          19
Holand-Netherlands                1
Name: native-country, dtype: int64
```

Kita mengingkan adanya penggabungan antara negara dengan region yang sama. Maka, kita akan membaginya menjadi 5 bagian yaitu `north-america`, `asia`, `south-america`, `europe`, dan `others`.

Terdapat 2 cara yang bisa digunakan, cara paling cepat untuk menghemat memory adalah sebagai berikut:

```
# buat dictionary
dict_countries = {
    'north_america' : ["Canada", "Cuba", "Dominican-Republic", "El-Salvador", "Guatemala", "Ha
                       "Jamaica", "Mexico", "Nicaragua","Outlying-US(Guam-USVI-etc)", "Puerto
                      "Trinadad&Tobago","United-States"],
    'asia' : ["Cambodia", "China", "Hong", "India", "Iran", "Japan", "Laos",
              "Philippines", "Taiwan", "Thailand", "Vietnam"],
    'south_america ' : ["Columbia", "Ecuador", "Peru"],
    'europe' : ["England", "France", "Germany", "Greece", "Holand-Netherlands","Hungary", "Irel
                    "Poland", "Portugal", "Scotland", "Yugoslavia"],
    'others' : ["South", "?"]
    }

# buat fungsi
def change_toregion(x):
    for k,v in dict_countries.items():
        if x in v:
            return k

# apply perubahan pada semua data
df_adult['native-country'] = df_adult['native-country'].apply(change_toregion)
```

Akan tetapi, kita akan menggunakan cara yang paling sederhana sebagai berikut

[18]:
```
# buat list untuk setiap region
north_america = ["Canada", "Cuba", "Dominican-Republic", "El-Salvador",␣
 ↪"Guatemala", "Haiti", "Honduras",
 "Jamaica", "Mexico", "Nicaragua","Outlying-US(Guam-USVI-etc)", "Puerto-Rico",
 "Trinadad&Tobago","United-States"]
asia = ["Cambodia", "China", "Hong", "India", "Iran", "Japan", "Laos",
          "Philippines", "Taiwan", "Thailand", "Vietnam"]
south_america = ["Columbia", "Ecuador", "Peru"]
europe = ["England", "France", "Germany", "Greece",␣
 ↪"Holand-Netherlands","Hungary", "Ireland",
          "Italy", "Poland", "Portugal", "Scotland", "Yugoslavia"]
others = ["South", "?"]
```

[19]:
```
# apply perubahan satu persatu
df_adult['native-country'] = df_adult['native-country'].apply(lambda x:␣
 ↪'north_america' if x in north_america else x)
df_adult['native-country'] = df_adult['native-country'].apply(lambda x: 'asia'␣
 ↪if x in asia else x)
```

```
df_adult['native-country'] = df_adult['native-country'].apply(lambda x:␣
 ↪'south_america' if x in south_america else x)
df_adult['native-country'] = df_adult['native-country'].apply(lambda x:␣
 ↪'europe' if x in europe else x)
df_adult['native-country'] = df_adult['native-country'].apply(lambda x:␣
 ↪'others' if x in others else x)
```

[20]:
```
# perhatikan perubahannya
df_adult['native-country'].value_counts()
```

[20]:
```
north_america    45933
asia               981
others             972
europe             780
south_america      176
Name: native-country, dtype: int64
```

1.1.4 Kolom Occupation

Kali ini kita hanya akan mengganti nilai ? menjadi `NaN` dengan cara yang sama di atas.

[21]:
```
df_adult.occupation.unique()
```

[21]:
```
array(['Machine-op-inspct', 'Farming-fishing', 'Protective-serv', '?',
       'Other-service', 'Prof-specialty', 'Craft-repair', 'Adm-clerical',
       'Exec-managerial', 'Tech-support', 'Sales', 'Priv-house-serv',
       'Transport-moving', 'Handlers-cleaners', 'Armed-Forces'],
      dtype=object)
```

[22]:
```
df_adult.occupation = df_adult.occupation.apply(lambda x: np.nan if x.
 ↪startswith('?') else x)
```

[23]:
```
df_adult.occupation.unique()
```

[23]:
```
array(['Machine-op-inspct', 'Farming-fishing', 'Protective-serv', nan,
       'Other-service', 'Prof-specialty', 'Craft-repair', 'Adm-clerical',
       'Exec-managerial', 'Tech-support', 'Sales', 'Priv-house-serv',
       'Transport-moving', 'Handlers-cleaners', 'Armed-Forces'],
      dtype=object)
```

1.2 2. Data Exploration

Tahap yang kedua ini sangat penting untuk dilakukan, umumnya kita melihat anomali yang terjadi di data dan melakukan tindakan yang diperlukan. Selanjutnya, tahapan berikut adalah melakukan visualisasi atas data yang ada.

1.2.1 Cari data anomali dan perbaiki

Saat ini kita memiliki banyak data NULL berupa `NaN`, langkah yang bisa kita lakukan adalah pengisian *(impute)* atau penghapusan. Pilihan pertama dapat kita lakukan apabila dirasa penting, namun langkah kedualah yang akan kita lakukan untuk kesederhanaan.

[24]:
```
len(df_adult[df_adult.occupation.isnull() == True])
```

[24]:
```
2809
```

[25]:
```
len(df_adult[df_adult.workclass.isnull() == True])
```

[25]:
```
2799
```

[26]:
```
len(df_adult[df_adult.workclass.isnull() & df_adult.occupation.isnull()])
```

[26]:
```
2799
```

[27]:
```
df_adult[df_adult.workclass.isnull() & df_adult.occupation.isnull()].head()
```

[27]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
4	18	NaN	103497	Some-college	10	Never-married	NaN	Own-child	White	Female	0	0	30	north_america	<=50K
6	29	NaN	227026	HS-grad	9	Never-married	NaN	Unmarried	Black	Male	0	0	40	north_america	<=50K
13	58	NaN	299831	HS-grad	9	Married	NaN	Husband	White	Male	0	0	35	north_america	<=50K
22	72	NaN	132015	7th-8th	4	Not-Married	NaN	Not-in-family	White	Female	0	0	6	north_america	<=50K
35	65	NaN	191846	HS-grad	9	Married	NaN	Husband	White	Male	0	0	40	north_america	<=50K

Dapat dilihat bahwa sebanyak 2.799 data terdapat 2 nilai `NaN` pada satu baris untuk kolom `occupation` dan `workclass`. Sedangkan, 2.809 hanya `occupation`.

Karena kita ingin memprediksi besaran `income` maka, kita harus memiliki nilai ini. Oleh karena itu, kita akan menghilangkan semua data `NaN` ini.

[28]:
```
# Hapus record yang memiliki nilai NaN pada attributnya
df_adult = df_adult.dropna(how='any')
```

[29]:
```
len(df_adult)
```

[29]:
```
46033
```

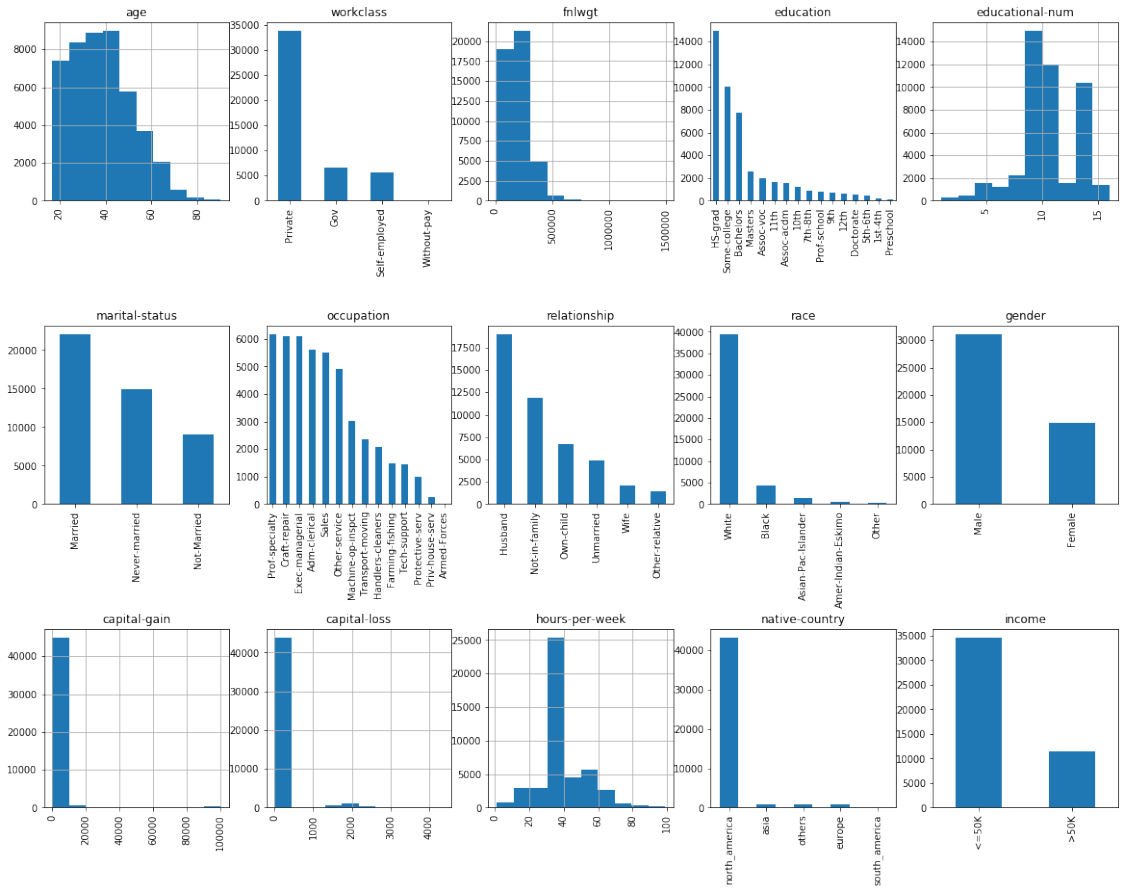
Sekarang, kita hanya memiliki 46.033 dataset setelah dibersihkan.

1.2.2 Cari Informasi Sebanyak-banyaknya dari data

Tahapan ini dapat kita sebut dengan **Data Exploration**, salah satu tools yang kita gunakan adalah **visualisasi**. Teknik ini menjadi penting karena kita bisa mendapatkan informasi yang sebelumnya tidak bisa didapatkan hanya dari angka dan kata.

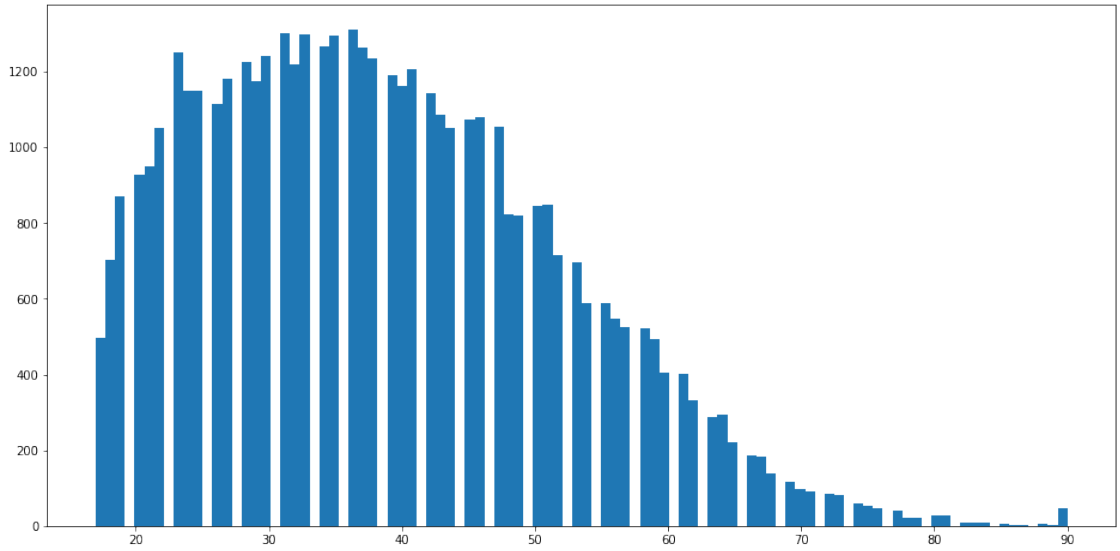
Berikut ini kita tampilkan plot informasi pada setiap kolom terhadap income

[30]:
```
from math import ceil

fig = plt.figure(figsize=(20,15))
cols = 5
rows = ceil(float(df_adult.shape[1]) / cols)
# looping setiap kolom dan indeksnya
for i, column in enumerate(df_adult.columns):
    # urutan setiap subplot
    ax = fig.add_subplot(rows, cols, i + 1)
    # set judul chart
    ax.set_title(column)

    if df_adult.dtypes[column] == np.object:
        df_adult[column].value_counts().plot(kind="bar", axes=ax)
    else:
        df_adult[column].hist(axes=ax)
        plt.xticks(rotation="vertical")

plt.subplots_adjust(hspace=0.7, wspace=0.2)
```



Dapat dilihat dari data di atas, sebagian besar data ada di daerah `north_america`, dan sebagian besar income di bawah $50,000. Di samping itu, ras yang mendominasi adalah kulit putih dan gendernya pria.

[31]:
```
plt.figure(figsize=(16, 8))
plt.hist(df_adult.age, bins=100, normed=None, histtype='bar', stacked=True)
plt.show()
```



Berikutnya kita akan melakukan visualisasi dengan menggunakan *heatmap*, unutk mendapatkan korelasi antara variabel. Akan tetapi, sebelumnya kita akan melakukan `label encoding` pada target variabel dan `gender`, karena mereka adalah binary variabel.

[32]:
```
salary_map={'<=50K': 1,'>50K':0}

df_adult.income = df_adult['income'].map(salary_map).astype(int)

df_adult['gender'] = df_adult['gender'].map({'Male':1,'Female':0}).astype(int)
```

Di samping itu, terdapat data categorical yaitu `education` yang akan kita lakukan encoding, karena terlihat sangat mirip dengan `educational-num`. Untuk itu, kita coba lakukan `LabelEncoder` pada fitur ini.

[33]:
```
df_adult.education.unique()
```

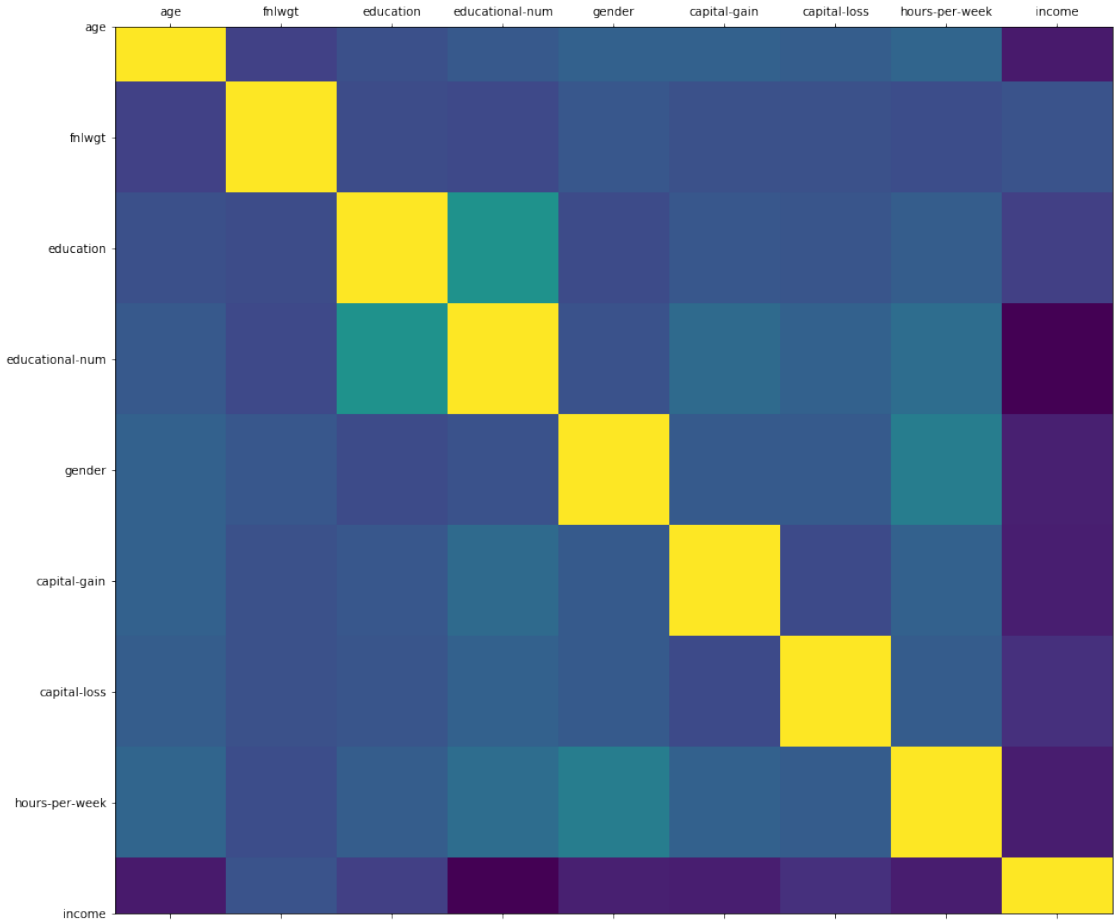
[33]:
```
array(['11th', 'HS-grad', 'Assoc-acdm', 'Some-college', '10th',
       'Prof-school', '7th-8th', 'Bachelors', 'Masters', 'Doctorate',
       '5th-6th', 'Assoc-voc', '9th', '12th', '1st-4th', 'Preschool'],
      dtype=object)
```

[34]:
```
from sklearn.preprocessing import LabelEncoder

df_adult.education = LabelEncoder().fit_transform(df_adult.education)
```

1.2.3 Visualisasi heatmap untuk mencari relasi variabel

Proses ini penting, terutama untuk menghilangkan variabel yang sangat berkorelasi.

[35]:
```
def plot_correlation(df, size=15):
    corr= df.corr()
    fig, ax =plt.subplots(figsize=(size,size))
    ax.matshow(corr)
    plt.xticks(range(len(corr.columns)),corr.columns)
    plt.yticks(range(len(corr.columns)),corr.columns)
    plt.show()

plot_correlation(df_adult)
```



Dari plot di atas terlihat dengan jelas bahwa `education` dan `educational-num` memang sangat berkaitan. Maka kita akan buang salah satu, tetapi `educational-num` ternyata memiliki properti **ordinal**, yaitu berurutan dengan makna dimana angka lebih besar, memiliki tingkat edukasi yang lebih tinggi.

[36]:
```
del df_adult['education']
```

1.2.4 Encode Categorical Data

Tahapan ini merupakan tahap akhir sebelum dilakukan data analisis.

Seluruh fitur data yang bersifat categorical string, harus diubah menjadi angka. Terdapat 2 cara yang umum dilakukan, yaitu cara manual dengan mengaplikasikan fungsi `replace`, `map`, `apply` dan lainnya, serta cara lain yaitu menggunakan fungsi `LabelEncoder`.

[37]:
```
df_adult.columns
```

[37]:
```
Index(['age', 'workclass', 'fnlwgt', 'educational-num', 'marital-status',
       'occupation', 'relationship', 'race', 'gender', 'capital-gain',
       'capital-loss', 'hours-per-week', 'native-country', 'income'],
      dtype='object')
```

[38]:
```
df_adult['native-country'] = LabelEncoder().
 ↪fit_transform(df_adult['native-country'])
df_adult['marital-status'] = LabelEncoder().
 ↪fit_transform(df_adult['marital-status'])
df_adult['relationship'] = LabelEncoder().
 ↪fit_transform(df_adult['relationship'])
df_adult['race'] = LabelEncoder().fit_transform(df_adult['race'])
df_adult['occupation'] = LabelEncoder().fit_transform(df_adult['occupation'])
df_adult['workclass'] = LabelEncoder().fit_transform(df_adult['workclass'])
```

1.3 3. Data Analisis

Tahap ini merupakan tahapan pembentukan model dari dataset yang ada. Kali ini, Anda akan diminta untuk melakukan *model selection* yang artinya, melakukan pemilihan model yang paling bagus di antara model-model yang ada.

Sebagaimana biasa, kita akan melakukan `train_test_split`. Kali ini, kita akan memisahkan antara data training dan data validasi.

[39]:
```
df = df_adult.copy()
```

[40]:
```
from sklearn.model_selection import train_test_split

# X merupakan dataset kecuali target variabel
X = df.drop(['income'],axis=1)
y = df.income

# kita gunakan 70% data training
split_size=0.3

# Creation of Train and Test dataset
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(X,y,test_size=split_size,random_state=22)

# Creation of Train and validation dataset
X_train, X_val, y_train, y_val = train_test_split(X_train,y_train,test_size=0.
 ↪2,random_state=5)
```

Import semua library classifier yang akan kita gunakan.

[41]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

[42]:
```
models = []

names = ['LR','Random Forest','Neural␣
 ↪Network','GaussianNB','DecisionTreeClassifier','SVM',]

models.append((LogisticRegression()))
models.append((RandomForestClassifier(n_estimators=100)))

models.append((MLPClassifier()))
models.append((GaussianNB()))
models.append((DecisionTreeClassifier()))
models.append((SVC()))
```

[43]:
```
from sklearn import model_selection
from sklearn.metrics import accuracy_score
```

Lakukan model selection dengan menggunakan KFold cross validation sebanyak 5 split dan hitung nilai accuracy nya.

[44]:
```
kfold = model_selection.KFold(n_splits=5,random_state=7)

for i in range(0,len(models)):
    cv_result = model_selection.
 ↪cross_val_score(models[i],X_train,y_train,cv=kfold,scoring='accuracy')
    score=models[i].fit(X_train,y_train)
    prediction = models[i].predict(X_val)
    acc_score = accuracy_score(y_val,prediction)
    print ('-'*40)
    print ('{0}: {1}'.format(names[i],acc_score))
```

```
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
```

```
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

----------------------------------------
LR: 0.7942591155934833
----------------------------------------
Random Forest: 0.8499612102404965

/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
```

```
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

----------------------------------------
Neural Network: 0.7976726144297905
----------------------------------------
GaussianNB: 0.7919317300232739
----------------------------------------
DecisionTreeClassifier: 0.8068269976726145

/Users/wahyuagungs/anaconda/lib/python3.6/site-packages/sklearn/svm/base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-packages/sklearn/svm/base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-packages/sklearn/svm/base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-packages/sklearn/svm/base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-packages/sklearn/svm/base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/wahyuagungs/anaconda/lib/python3.6/site-packages/sklearn/svm/base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)

----------------------------------------
SVM: 0.7627618308766486
```

Dari Data di atas ternyata, `RandomForestClassfier` memiliki nilai akurasi tertinggi sebesar 85%.

1.4 4. Evaluasi atas performa model

Dikarenakan `RandomForestClassifier` memiliki nilai tertinggi, kita akan menggunakan model tersebut sebagai baseline untuk pengukuran performa model kita.

[45]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

[46]:
```
randomForest = RandomForestClassifier(n_estimators=100)
randomForest.fit(X_train,y_train)
prediction = randomForest.predict(X_test)
```

[47]:
```
print ('-'*40)
print ('Accuracy score:')
print (accuracy_score(y_test,prediction))
print ('-'*40)
print ('Confusion Matrix:')
print (confusion_matrix(y_test,prediction))
print ('-'*40)
print ('Classification Matrix:')
print (classification_report(y_test,prediction))
```

```
----------------------------------------
Accuracy score:
0.8547429398986242
----------------------------------------
Confusion Matrix:
[[2191 1281]
 [ 725 9613]]
----------------------------------------
Classification Matrix:
              precision    recall  f1-score   support

           0       0.75      0.63      0.69      3472
           1       0.88      0.93      0.91     10338

    accuracy                           0.85     13810
   macro avg       0.82      0.78      0.80     13810
weighted avg       0.85      0.85      0.85     13810
```

Dapat diketahui nilai sebagai berikut:

- TP = 2184
- TN = 9629
- FN = 909
- FP = 1288

Recall = 70.6% Precision = 62.9% Accuracy = 85%