Python for Data Science

Ch. 4
Data Cleaning

Ahmad Rio Adriansyah

Data yang Bagus

- Data akan lebih bagus dan kualitasnya lebih tinggi jika diperbaiki dari sumbernya
- Setelah diperbaiki dari sumbernya, kita dapat melakukan
  - **Standardisasi**, untuk memastikan tipe data di tiap kolom seragam
  - **Normalisasi**, untuk memastikan konsistensi semua data yang ada
  - **Merging datasets**, jika data berada di beberapa dataset yang terpisah
  - **Agregasi**, untuk mengurutkan dan mengelompokkan data dengan aturan tertentu
  - **Filtering**, untuk membatasi dataset ke informasi yang dibutuhkan saja

Merging Dataframe

- Jika ada data yang terpisah pada 2 dataset, ingin digabungkan pada sebuah dataset
- Dapat menggabungkan berdasarkan baris atau kolom dengan **concat()** atau **merge()**
- Menggabungkan baris

```
df1 = pd.read_csv('karyawan1.csv')
df2 = pd.read_csv('karyawan2.csv')
```

```
df = pd.concat([df1,df2])
```

- Menggabungkan kolom

```
dfx = pd.concat([df1,df2],axis = 1)
```

Merging Dataframe

- Merge prinsipnya sama seperti JOIN pada SQL
- Menggabungkan 2 tabel data berdasarkan kriteria tertentu

```
nilai_2018 = pd.read_csv('karyawan-penilaian2018.csv')
```

```
dfnilai = pd.merge(df,nilai_2018, on=['ID','Nama Pegawai'])
```

Merging Dataframe

- Jika header di dataframe berbeda, dapat digunakan parameter **left_on** dan **right_on**

```
nilai_2019 = pd.read_csv('karyawan-penilaian2019.csv')
```

```
dfnilai = pd.merge(df, nilai_2019, left_on = ['ID','Nama Pegawai'],
                   right_on = ['id_karyawan','nama_pegawai'] )
```

Merging Dataframe

- **Full outer join** menghasilkan semua baris pada data frame pertama dan kedua. Jika tidak ada yang cocok (match), maka bagian yang tidak ada akan diberi nilai null

```
pd.merge(df, nilai_2018, on = ['ID','Nama Pegawai'], how = 'outer')
```

- **Inner join** hanya menghasilkan baris yang cocok pada data frame pertama dan kedua.

```
pd.merge(df, nilai_2018, on = ['ID','Nama Pegawai'], how = 'inner')
```

Merging Dataframe

- **Left join** dan **right join** juga bisa dilakukan

```
pd.merge(df, nilai_2018, on = ['ID','Nama Pegawai'], how = 'left')
pd.merge(df, nilai_2018, on = ['ID','Nama Pegawai'], how = 'right')
```

- Kita juga dapat menambahkan akhiran (**suffixes**) tertentu untuk membedakan kolom dari data frame pertama dan kedua

```
pd.merge(df, nilai_2018, on = ['ID','Nama Pegawai'],
         suffixes =('_karyawan','_nilai'))
```

Agregasi

- Untuk mengagregasi data frame terhadap nilai tertentu dapat dilakukan menggunakan fungsi **groupby()**
- Dataset yang akan kita gunakan
  - https://www.kaggle.com/karangadiya/fifa19
  - https://www.kaggle.com/abecklas/fifa-world-cup

Agregasi

- Misalnya kita ingin menggabungkan data tersebut berdasarkan kewarganegaraannya

```
dffifa = pd.read_csv('fifa19.csv')
```

```
nasional = dffifa.groupby('Nationality')
```

- Fungsi **groupby()** menghasilkan sebuah objek **DataFrameGroupBy** , bukan **DataFrame**

Group By

- Melihat anggota pertama dari tiap grup

```
nasional.first()
```

- Melihat grup yang ada

```
nasional.groups.keys()
```

Melihat seluruh anggota dari grup tertentu

```
nasional.get_group('Brazil')
```

Group Summary

- Pada DataFrameGroupBy dapat diaplikasikan perintah tambahan untuk mendapatkan hasil summarynya.
  - Sum()
  - Count()
  - Max()
  - Min()
  - Mean()
  - dll
- Hasil summarynya akan berupa data frame

Latihan

- Kelompokkan data pemain sepakbola berdasarkan klubnya
- Hitung jumlah harga pemain tiap klub
- Klub mana yang jumlah harga pemainnya paling tinggi?
- Bagaimana dengan klub yang rata-rata harga pemainnya paling tinggi? Klub manakah itu?

Sorting

- Untuk mengurutkan sebuah data frame digunakan fungsi **sort_values()**

```
df.sort_values('ID')
```