Universitas Indonesia

Tugas 1: DATA PREPARATION

PENAMBANGAN DATA DAN INTELEGENSIA BISNIS

BAHY HELMI HARTOYO PUTRA
1606918124

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI SISTEM INFORMASI
DEPOK
OKTOBER 2019

Pendahuluan

Data preparation merupakan hal yang penting untuk dilakukan sebelum memulai sebuah *data science*/*machine learning project*. Terkadang kita berfikir bahwa Data Scientist banyak menghabiskan waktunya pada proses perancangan algoritma *machine learning* (*modelling*). Namun realitanya, kebanyakan Data Scientist menghabiskan waktunya untuk menjalankan proses *data preparation*.

Tugas ini akan mencakup proses awal dari sebuah *machine learning/data science project*, yaitu *data preparation. Data preparation* akan dilaksanakan menggunakan beberapa teknik yang telah dipelajari seperti eksplorasi, transformasi, *smoothing, outlier analysis,* dan *imputation*.

Deskripsi Data

Sebelum melakukan operasi pada sebuah data, sangat penting bagi seorang Data Scientist untuk dapat memahami data yang dimiliki. Selain itu, pengetahuan tentang apa yang ingin dicapai menggunakan data tersebut juga harus dimilki. Pemahaman akan dua hal ini mempemiudah seorang Data Scientist dalam melakukan pengambilan keputusan saat menjalankan proses *data preparation*.

Data yang akan diolah adalah *data-t1.csv* yang tersedia pada deskripsi Tugas 1. Berikut merupakan deskripsi mengenai data tersebut:

Nama Variabel	Tipe Data	Penjelasan
loan_status	(str, categorical)	Variabel degan *multiple levels* yang menandakan status pinjaman seseorang (e.g. Charged off, Current, Default, Fully Paid, etc)
loan_amnt	(int, discrete)	Variabel yang berisikan jumlah pinjaman yang dimilki seseorang (e.g. 1000, 2000, 3000, etc.)
int_rate	(float, continuous)	Variabel yang berisikan bunga pinjaman seseorang (e.g. 13.56, 2.5, 12.3, etc.)
grade	(str, categorical)	Variabel yang berisikan tingkat *employment* seseorang (e.g. A, B, C, etc.)
emp_length	(str, categorical)	Variabel yang berisikan

		durasi *employment* seseorang (e.g. 4 years, 5 years, etc.)
home_ownership	(str, categorical)	Variabel yang berisikan status kepemilikan rumah seseorang (e.g. RENT, ANY, etc.)
annual_inc	(float, continuous)	Variabel yang berisikan total penghasilan tahunan seseorang (e.g. 111.24, 123.14, etc.)
term	(str, categorical)	Variabel yang berisikan term seseorang (e.g. 60 months, 36 months)

Data awal berisikan 149997 baris dan 8 kolom.

Data Preparation

Data preparation merupakan sebuah proses untuk melakukan persiapan terhadap sebuah *raw data*. Raw data yang baru di*mining*/diterima Dalam melakukan *data preparation* kali ini, akan ada beberapa tahapan yang dilakukan, yaitu sebagai berikut:

- **Load Data**: melakukan *load* terhadap data yang ingin diolah, baik bersumber sebuah *cloud storage* (GCS/BigQuery, AWS) ataupun pada kasus ini dari data *local* yaitu sebuah file *csv*.
- **Eksploration & Data Checking**: pada tahapan ini yang dilakukan adalah melakukan *sanity check* terhadap data dan melihat *errors* yant teradapat pada data. Errors dapat merupakan *miss-recorded* data ataupun *null values* data.
- **Data Smoothing:** terkadang perbedaan antara data yang minor tidak terlalu signifikan dan tidak menurunkan performa. Data *smoothing* dapat membuat data-data yang memiliki perbedaan tidak signifikan tersebut menjadi lebih seragam.
- **Data Imputation:** *imputation* dilakukan untuk melakukan *replacement* terhadap *null values* yang ada pada suatu kolom. Teknik *imputation* ada berbagai macam, dapat *berupa mean/median/mode imputation*.
- **Data Transformation:** data terkadang perlu untuk dilakukan transformasi. Hal ini membuat sebuah data menjadi lebih *usable*, lebih mudah diinterpretasikan, serta *useful* untuk digunakan pada tahapan berikutnya.
- **Outliers Treatment:** *outliers* dapat membuat data salah diinterpretasikan saat melihat statistik dari data tersebut. Sekumpulan atau sebuah *outlier* dapat menarik *mean* ke arah negatif/positif yang mana sebenarnya pesebaran distribusi data tidak berada di titik tersebut. Outliers perlu ditangani dengan cara dihilangkan.

Proses *data preparation* dilakukan menggunakan bantuan bahasa pemrograman Python pada sebuah tools, yaitu Jupyter Notebook. Library yang digunakan pada *data preparation* kali ini adalah NumPy, pandas, dan juga seaborn. Detil pengerjaan dan *notebook* dapat diakses pada lampiran.

Hasil/Temuan

- Terdapat **2 *bad lines*** pada *raw* data, yaitu pada line 52431 dan 131201. Seharusnya tabel ini hanya memiliki 8 kolom, namun pada baris tersebut data tersebar pada 9 kolom.
- Data *raw* tanpa *bad lines* memiliki **149997 baris** dan **8 kolom**.
- *Miss-recorded data*, di luar *null values*, terdapat pada kolom-kolom:
    - *loan_status* = 1213 baris
    - *emp_length* = 35 baris
    - *home_ownership* = 5 baris
- ***Null values** data* terdapat pada kolom *emp_length*, yaitu sebanyak **9%** dari data atau sebanyak **13668** baris.
- ***Smoothing*** perlu dilakukan pada kolom *annual_inc*, *smoothing* dilakukan pada data yang bertipe *float* pada kolom tersebut. Operasi ***round*** dilakukan untuk mengubah ke *nearest integer*.
- **Imputation** dilakukan pada kolom emp_length dimana terdapat 9% *null values*. Karena raw data bertipe *qualitative* (*categorical*), maka ***mode imputation*** dilakukan untuk mengisi *null values.*
- **Transformasi** dilakukan kepada kolom-kolom yang bertipe ordinal dan nominal.
    - Kolom yang bertipe **ordinal** (*grade* & *emp_length*) ditransform ke dalam numerical. Grade A, G melambangkan tingkatan *risk* yang dimiliki, A paling kecil, G paling besar, sehingga dapat di*transform* menjadi angka 1-7. Employment length dapat ditransform menjadi angka 0-10, dimana 0 melambangkan *employment length* dibawah 1 tahun, dan 10 merupakan lebih dari 10 tahun.
    - Kolom yang bertipe **nominal** (*loan_status, home_ownership, term*) ditransform dengan bantuan pandas.get_dummies untuk menjadikan sebagai kolom *numerical boolean* (1,0) dengan membuat *value* menjadi judul kolom.
- **Outliers treeatment** perlu dilakukan untuk kolom *annual_inc* dan *int_rate*. Treatment yang dilakukan adalah melakukan *outliers removement* berdasarkan threshold yang telah ditentukan. Penentuan *threshold* dilakukan berdasarkan aturan berikut:
    - Q1 - 1.5 IQR, Q3 + 1.5 IQR (diterapkan pada kolom *annual_inc*)
    - Mean +- 2 Standard Deviation (diterapkan pada kolom *int_rate*)
- Data final yang telah siap untuk digunakan pada proses berikutnya berisikan **128548 baris** dan **17 kolom.**

Kesimpulan

Dari proses *data preparation* yang dilakukan, ada beberapa poin-poin penting yang dapat disimpulkan, yaitu:

- Melakukan *data preparation* sebelum memulai sebuah *machine learning project* sangatlah penting. Data tidak dapat langsung digunakan karena data memiliki resiko mengandung *error/miss-recorded* data.
- Memahami teknik *data preparation* sangatlah penting, dengan pemahaman tersebut kita akan lebih mudah untuk melakukan operasi-operasi yang diinginkan pada data yang dimilki.
- Memahami konteks serta setiap variabel yang ada dalam data yang dimiliki juga penting. Tanpa mengetahui hal tersebut, validasi terhadap data akan sulit untuk dilakukan. Pemahaman ini juga akan membantu dalam proses *brainstorming* saat ingin memulai *data preparation*.

Lampiran

Notebook dan hasil dari data preparation dapat diakses pada link berikut:
http://bit.ly/Tugas1-PDIB-Bahy

Data Preparation

Bahy Helmi Hartoyo Putra
1606918124
Tugas 1 - PDIB

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

1. Load Data

In [2]:
```
## Load data dari CSV
df = pd.read_csv("data-t1.csv")
```

```
---------------------------------------------------------------------------
ParserError                               Traceback (most recent call last)
<ipython-input-2-5f70fca2747d> in <module>
      1 ## Load data dari CSV
----> 2 df = pd.read_csv("data-t1.csv")

c:\users\bahyh\appdata\local\programs\python\python37-32\lib\site-packages\pandas\io\parsers.py in parser_
f(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_col
s, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na
_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_d
ate_col, date_parser, dayfirst, iterator, chunksize, compression, thousands, decimal, lineterminator, quot
echar, quoting, doublequote, escapechar, comment, encoding, dialect, tupleize_cols, error_bad_lines, warn_
bad_lines, delim_whitespace, low_memory, memory_map, float_precision)
    700                     skip_blank_lines=skip_blank_lines)
    701
--> 702         return _read(filepath_or_buffer, kwds)
    703
    704     parser_f.__name__ = name

c:\users\bahyh\appdata\local\programs\python\python37-32\lib\site-packages\pandas\io\parsers.py in _read(f
ilepath_or_buffer, kwds)
    433
    434     try:
--> 435         data = parser.read(nrows)
    436     finally:
    437         parser.close()

c:\users\bahyh\appdata\local\programs\python\python37-32\lib\site-packages\pandas\io\parsers.py in read(se
lf, nrows)
   1137     def read(self, nrows=None):
   1138         nrows = _validate_integer('nrows', nrows)
-> 1139         ret = self._engine.read(nrows)
   1140
   1141         # May alter columns / col_dict

c:\users\bahyh\appdata\local\programs\python\python37-32\lib\site-packages\pandas\io\parsers.py in read(se
lf, nrows)
   1993     def read(self, nrows=None):
   1994         try:
-> 1995             data = self._reader.read(nrows)
   1996         except StopIteration:
   1997             if self._first_chunk:

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader.read()

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._read_low_memory()

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._read_rows()

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._tokenize_rows()

pandas\_libs\parsers.pyx in pandas._libs.parsers.raise_parser_error()

ParserError: Error tokenizing data. C error: Expected 8 fields in line 52431, saw 9
```

Terdapat bad lines pada line 52431 dan 131201, dimana row tersebut memiliki 9 kolom, lebih banyak 1 kolom dibanding row lainnya. Maka kita dapat melakukan *skipping* untuk dua row tersebut.

In [3]:
```
## Load data, skip error/bad lines
df = pd.read_csv("data-t1.csv", error_bad_lines=False)
```

```
b'Skipping line 52431: expected 8 fields, saw 9\n'
b'Skipping line 131201: expected 8 fields, saw 9\n'
```

Line 52431 dan 131201 memiliki lebih dari 8 kolom, kedua line tersebut dapat dilewati (disingkirkan dari tabel).

In [4]:
```
## Melakukan checking terhadap dataset yang dimiliki
df.shape
```

Out[4]: (149997, 8)

In [5]:
```
## Memastikan index yang telah diskip tidak ada dalam dataframe (+-52431)
df.iloc[52429:].head()
```

Out[5]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
52429	Current	8000	13.56	C	2 years	MORTGAGE	80000	36 months
52430	In Grace Period	12000	10.72	B	10+ years	MORTGAGE	140000	36 months
52431	Current	12000	19.92	D	NaN	RENT	0	60 months
52432	Current	8000	7.56	A	10+ years	MORTGAGE	155000	36 months
52433	Current	4600	14.47	C	1 year	RENT	18430	36 months

In [6]:
```
## Memastikan index yang telah diskip tidak ada dalam dataframe (+-131201)
df.iloc[131199:].head()
```

Out[6]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
131199	Current	18000	20.89	D	1 year	OWN	83000	36 months
131200	Current	20000	11.55	B	5 years	MORTGAGE	120000	60 months
131201	Current	10000	16.14	C	1 year	RENT	60000	36 months
131202	Current	30000	12.73	B	4 years	MORTGAGE	100000	60 months
131203	Fully Paid	13000	16.91	C	10+ years	MORTGAGE	55000	36 months

2. Examine Data

Sanity check pada data awal

In [7]:
```
## 5 baris pertama dari data
df.head()
```

Out[7]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
0	Current	2500	13.56	C	10+ years	RENT	55000	36 months
1	Current	30000	18.94	D	10+ years	MORTGAGE	90000	60 months
2	Current	5000	17.97	D	6 years	MORTGAGE	59280	36 months
3	Current	4000	18.94	D	10+ years	MORTGAGE	92000	36 months
4	Current	30000	16.14	C	10+ years	MORTGAGE	57250	60 months

In [8]:
```
## Cek shape dari data
df.shape
```

Out[8]: (149997, 8)

Data memiliki 149997 baris dan 8 kolom.

In [9]:
```
df.head(1)
```

Out[9]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
0	Current	2500	13.56	C	10+ years	RENT	55000	36 months

Berdasarkan referensi berikut (http://wcw.cs.ui.ac.id/teaching/imgs/bahan/pdib/data-t1.txt), baris di atas merupakan contoh dari sebuah baris yang valid/sesuai.

2.1 Check for "Miss-recorded"

Expected values/data types pada setiap kolom yang ada adalah sebagai berikut:

- **loan_status**: (str, categorical) (e.g. Charged off, Current, Default, Fully Paid, etc.)
- **loan_amnt**: (int, discrete) (e.g. 1000, 2000, 3000, etc.)
- **int_rate**: (float, continuous) (e.g. 13.56, 2.5, 12.3, etc.)
- **grade** (str, categorical) (e.g. A, B, C, etc.)
- **emp_length** (str, categorical) (e.g. 4 years, 5 years, etc.)
- **home_ownership** (str, categorical) (e.g. RENT, MORTGAGE, etc.)
- **annual_inc** (float, continuous) (e.g. 111.24, 1231.4, etc.)
- **term** (str, categorical) (e.g. 60 months, 36 months, etc.)

Jika terdapat sebuah baris yang mengandung tipe data/value yang tidak sesuai, maka baris tersebut dapat didrop.

Loan Status

In [10]:
```
## Check value yang sekarang ada pada laon_status
df['loan_status'].unique()
```

Out[10]:
```
array(['Current', 'Fully Paid', 'Dec-2018', 'Late (31-120 days)',
       'In Grace Period', 'Charged Off', 'Verified', 'Late (16-30 days)',
       'Not Verified', 'Source Verified', 'Nov-2018', 'Oct-2018', 'RENT',
       '32000', 'Sep-2018', 'Fulli Paid', 'Full Paid', 'Curren', 'Curent'],
      dtype=object)
```

In [11]:
```
## Check jumlah dari setiap value yang ada pada loan_status
df.groupby(['loan_status']).size()
```

Out[11]:
```
loan_status
32000                    1
Charged Off             73
Curent                   1
Curren                   2
Current             142877
Dec-2018               303
Full Paid                1
Fulli Paid               1
Fully Paid            4272
In Grace Period        592
Late (16-30 days)      232
Late (31-120 days)     704
Not Verified            20
Nov-2018               347
Oct-2018               375
RENT                     1
Sep-2018               181
Source Verified         10
Verified                 4
dtype: int64
```

In [12]:
```
## Status valid yang seharusnya ada pada kolom loan_status
valid_status = ['Charged Off', 'Current', 'Fully Paid', 'In Grace Period', 'Late (16-30 days)', 'Late (31-
120 days)',\
                'Not Verified', 'Source Verified', 'Verified', np.nan]
```

In [13]:
```
## Update data dengan kondisi dimana isi dari loan_status hanya merupakan status yang valid
df = df[df['loan_status'].isin(valid_status)]
```

Status yang tidak valid dapat langsung diremove, karena memang beberapa data yang berupa **miss-recorded** data (e.g. Dec-2018, Nov-2018, dsb.) bukan merupakan data yang bisa diolah. Data-data yang **typo** (e.g. Curent, Fulli Paid) dapat diabaikan karena jumlahnya sangat sedikit.

Loan Amount

In [14]:
```
## Check jumlah data pada kolom loan_amnt yang merupakan integer
df['loan_amnt'].astype(str).str.isdigit().sum()
```

Out[14]: 148784

In [15]:
```
## Check data pada kolom loan_amnt yang bukan merupakan integer
df[~df['loan_amnt'].astype(str).str.isdigit()]
```

Out[15]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term

In [16]:
```
df['loan_amnt'].describe()
```

Out[16]:
```
count    148784.000000
mean      15979.620288
std       10117.444912
min        1000.000000
25%        8000.000000
50%       14000.000000
75%       21600.000000
max       40000.000000
Name: loan_amnt, dtype: float64
```

Semua data pada kolom loan_amnt telah berisikan **integer** dengan nilai minimum 1000 dan maksimum 40000.

Interest Rate

In [17]:
```
## Check tipe data yang terdapat pada kolom int_rate
df['int_rate'].dtype
```

Out[17]: dtype('float64')

In [18]:
```
df['int_rate'].describe()
```

Out[18]:
```
count    148784.000000
mean         12.913209
std           5.127769
min           6.000000
25%           8.460000
50%          11.800000
75%          16.140000
max          30.990000
Name: int_rate, dtype: float64
```

Semua data pada kolom int_rate telah berisikan **float** dengan nilai minimum 6 dan maksimum 30.99.

Grade

In [19]:
```
## Check tipe data yang terdapat pada kolom grade
df['grade'].unique()
```

Out[19]: array(['C', 'D', 'B', 'A', 'E', 'F', 'G'], dtype=object)

Semua data pada kolom grade telah sesuai, berisikan category A-G.

Employment Length

In [20]:
```
## Check values yang terdapat pada kolom emp_length
df['emp_length'].unique()
```

Out[20]:
```
array(['10+ years', '6 years', '4 years', '< 1 year', '2 years',
       '9 years', nan, '5 years', '3 years', '7 years', '1 year',
       '8 years', ' Policy', ' Office of Conferences', ' Team Leader',
       ' Windows', ' CSO', ' Governance', 'mason', ' Finance',
       'Purchasing Manager', ' GA', 'M', ' Assistant', ' Marketing',
       ' Cosmetic', ' equipment operator', ' NV', 'Hostess', ' Artist',
       ' Media Director', ' LVN', ' maintenance', 'bellman',
       ' Volunteer ; HR', ' Head Tech', ' 230 Appleton Pl', ' bartender',
       ' Physician', ' Bus driver', ' Benefits',
       ' Health Management Nurse', 'Cashier', ' hostess', ' operator',
       ' machinest'], dtype=object)
```

In [21]:
```
## Check jumlah dari setiap value yang ada pada kolom emp_length
df.groupby(['emp_length']).size()
```

Out[21]:
```
emp_length
 230 Appleton Pl                1
 Artist                         1
 Assistant                      1
 Benefits                       1
 Bus driver                     1
 CSO                            1
 Cosmetic                       1
 Finance                        1
 GA                             1
 Governance                     1
 Head Tech                      1
 Health Management Nurse        1
 LVN                            1
 Marketing                      1
 Media Director                 1
 NV                             1
 Office of Conferences          1
 Physician                      1
 Policy                         1
 Team Leader                    1
 Volunteer ; HR                 1
 Windows                        1
 bartender                      1
 equipment operator             1
 hostess                        1
 machinest                      1
 maintenance                    1
 operator                       1
1 year                      10395
10+ years                   45066
2 years                     12574
3 years                     11699
4 years                      8718
5 years                      8984
6 years                      6287
7 years                      5065
8 years                      4742
9 years                      3226
< 1 year                    18326
Cashier                         1
Hostess                         1
M                               1
Purchasing Manager              1
bellman                         1
mason                           1
dtype: int64
```

In [22]:
```
## Length valid yang seharusnya ada pada kolom emp_length
valid_length = ['10+ years', '6 years', '4 years', '< 1 year', '2 years','9 years', \
                np.nan, '5 years', '3 years', '7 years', '1 year', '8 years']
```

Null values masih di-*include*, baru akan diremove pada bab berikutnya (**Imputation**).

In [23]:
```
## Update data dengan kondisi dimana isi dari emp_length hanya merupakan length yang valid
df = df[df['emp_length'].isin(valid_length)]
```

In [24]:
```
## Check jumlah dari setiap value yang ada pada kolom emp_length
df['emp_length'].unique()
```

Out[24]:
```
array(['10+ years', '6 years', '4 years', '< 1 year', '2 years',
       '9 years', nan, '5 years', '3 years', '7 years', '1 year',
       '8 years'], dtype=object)
```

Semua data pada kolom emp_length telah sesuai, berisikan category < 1 year - 10+ years.

Home Ownership

In [25]:
```
## Check values yang terdapat pada kolom home_ownership
df['home_ownership'].unique()
```

Out[25]:
```
array(['RENT', 'MORTGAGE', 'OWN', 'ANY', 'MOTGAGE', 'MORGAGE'],
      dtype=object)
```

In [26]:
```
## Check jumlah dari setiap value yang ada pada kolom home_ownership
df.groupby(['home_ownership']).size()
```

Out[26]:
```
home_ownership
ANY           363
MORGAGE         2
MORTGAGE    73413
MOTGAGE         3
OWN         16388
RENT        58581
dtype: int64
```

In [27]:
```
## Ownership valid yang seharusnya ada pada kolom home_ownership
valid_ownership = ['RENT', 'MORTGAGE', 'OWN', 'ANY', np.nan]
```

In [28]:
```
## Update data dengan kondisi dimana isi dari home_ownership hanya merupakan length yang valid
df = df[df['home_ownership'].isin(valid_ownership)]
```

Semua data pada kolom home_ownership telah sesuai, berisikan category RENT, MORGAGE, OWN, ANY.

Annual Income

In [29]:
```
## Check jumlah data pada kolom annual_inc yang merupakan integer
df['annual_inc'].str.isdigit().sum()
```

Out[29]:
```
148109
```

In [30]:
```
## Check data pada kolom annual_inc yang bukan merupakan integer
df[~df['annual_inc'].str.isdigit()][['annual_inc']].head()
```

Out[30]:

	annual_inc
1115	70775.28
1924	19110.59
2450	82492.8
2572	63659.88
2631	33865.68

In [31]:
```
## Memastikan data selain integer hanya merupakan data float
df[~df['annual_inc'].str.isdigit()]['annual_inc'].astype(float).head()
```

Out[31]:
```
1115    70775.28
1924    19110.59
2450    82492.80
2572    63659.88
2631    33865.68
Name: annual_inc, dtype: float64
```

Semua data pada kolom annual_inc berisikan float & integer, untuk saat ini cukup melakukan checking bahwa kolom tersebut mengandung data numerical. Data yang masih berbentuk float dapat ditangani dengan melakukan **smoothing** pada step berikutnya.

Term

In [32]:
```
## Check values yang ada pada kolom term
df['term'].unique()
```

Out[32]:
```
array([' 36 months', ' 60 months'], dtype=object)
```

Semua data pada kolom term telah sesuai, berisikan dua category term yaitu 36 months dan 60 months.

2.2 Check for "Null Values"

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 148745 entries, 0 to 149996
Data columns (total 8 columns):
loan_status       148745 non-null object
loan_amnt         148745 non-null int64
int_rate          148745 non-null float64
grade             148745 non-null object
emp_length        135077 non-null object
home_ownership    148745 non-null object
annual_inc        148745 non-null object
term              148745 non-null object
dtypes: float64(1), int64(1), object(6)
memory usage: 6.8+ MB
```

Hanya terdapat 1 kolom yang memiliki null values, yaitu **emp_length**.

In [34]:
```
## Jumlah null values pada kolom emp_length (%)
df['emp_length'].isnull().sum()/df.shape[0]*100
```

Out[34]:
```
9.188880298497429
```

In [35]:
```
## Jumlah null values pada kolom emp_length (row)
df['emp_length'].isnull().sum()
```

Out[35]:
```
13668
```

Ada 9% (13668 rows) pada kolom emp_length yang memiliki null values.

3. Smoothing, Imputation, Transformation, and Outlier Treatment

In [36]:
```
df.head()
```

Out[36]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
0	Current	2500	13.56	C	10+ years	RENT	55000	36 months
1	Current	30000	18.94	D	10+ years	MORTGAGE	90000	60 months
2	Current	5000	17.97	D	6 years	MORTGAGE	59280	36 months
3	Current	4000	18.94	D	10+ years	MORTGAGE	92000	36 months
4	Current	30000	16.14	C	10+ years	MORTGAGE	57250	60 months

3.1 Smoothing

- Sebagian data annual income masih berupa decimal (float), dapat dilakukan data smoothing untuk menghilangkan perbedaan yang tidak signifikan.
- Interest rate tidak perlu dilakukan *smoothing* karena 2 angka di belakang koma pada bunga dapat berperan penting dalam pengali suatu value.

In [37]:
```
## DataFrame dengan annual_inc yang memiliki tipe data selain integer (float)
df[~df['annual_inc'].astype(str).str.isdigit()].head()
```

Out[37]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
1115	Current	40000	16.14	C	10+ years	RENT	70775.28	60 months
1924	Fully Paid	1000	23.40	E	NaN	OWN	19110.59	36 months
2450	Current	20000	8.81	A	10+ years	MORTGAGE	82492.8	36 months
2572	Current	18000	6.46	A	10+ years	RENT	63659.88	36 months
2631	Current	3500	19.92	D	NaN	OWN	33865.68	36 months

In [38]:
```
## Simpan ke sebuah variable baru, untuk memisahkan annual income float & integer
df_ann_income_float = df[~df['annual_inc'].astype(str).str.isdigit()]
df_ann_income_int = df[df['annual_inc'].astype(str).str.isdigit()]
## Rounding float pada annual_inc ke integer
ann_income_float = df_ann_income_float['annual_inc'].apply(lambda x: round(float(x)))
## Assign annual income yang sudah berbentuk integer ke dataframe
df_ann_income_float['annual_inc'] = ann_income_float
```

```
c:\users\bahyh\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:7: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-v
iew-versus-copy
  import sys
```

In [39]:
```
## Float telah diround ke nearest integer
df_ann_income_float.head()
```

Out[39]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
1115	Current	40000	16.14	C	10+ years	RENT	70775	60 months
1924	Fully Paid	1000	23.40	E	NaN	OWN	19111	36 months
2450	Current	20000	8.81	A	10+ years	MORTGAGE	82493	36 months
2572	Current	18000	6.46	A	10+ years	RENT	63660	36 months
2631	Current	3500	19.92	D	NaN	OWN	33866	36 months

In [40]:
```
## Update tabel dengan data yang telah di smoothing
df = pd.concat([df_ann_income_float, df_ann_income_int]).sort_index()
df['annual_inc'] = df['annual_inc'].astype(int)
```

3.2 Imputation

Karena data pada kolom **emp_length** merupakan qualitative data, maka yang akan digunakan adalah *mode imputation*.

In [41]:
```
df.groupby(['emp_length']).size().sort_values(ascending=False)
```

Out[41]:
```
emp_length
10+ years    45065
< 1 year     18324
2 years      12574
3 years      11699
1 year       10395
5 years       8984
4 years       8717
6 years       6286
7 years       5065
8 years       4742
9 years       3226
dtype: int64
```

In [42]:
```
## DataFrame yang mengandung null values pada kolom emp_length
df[df['emp_length'].isnull()].head()
```

Out[42]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
25	Current	15000	14.47	C	NaN	MORTGAGE	30000	60 months
34	Current	20000	11.80	B	NaN	MORTGAGE	47590	60 months
41	Current	2200	15.02	C	NaN	RENT	70000	36 months
43	Current	4000	11.80	B	NaN	RENT	20000	36 months
46	Current	7500	11.80	B	NaN	MORTGAGE	32700	36 months

In [43]:
```
## Simpan ke sebuah variable baru, untuk memisahkan employement length null dan yang tidak null
df_emp_length_null = df[df['emp_length'].isnull()]
df_emp_length_not_null = df[~df['emp_length'].isnull()]
## Assign 10+ years ke value-value yang tadinya null
df_emp_length_null['emp_length'] = '10+ years'
```

```
c:\users\bahyh\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:5: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-v
iew-versus-copy
  """
```

In [44]:
```
## Null values pada emp_length telah diisikan dengan mode dari kolom tersebut (10+ years)
df_emp_length_null.head()
```

Out[44]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
25	Current	15000	14.47	C	10+ years	MORTGAGE	30000	60 months
34	Current	20000	11.80	B	10+ years	MORTGAGE	47590	60 months
41	Current	2200	15.02	C	10+ years	RENT	70000	36 months
43	Current	4000	11.80	B	10+ years	RENT	20000	36 months
46	Current	7500	11.80	B	10+ years	MORTGAGE	32700	36 months

In [45]:
```
## Update tabel dengan data yang telah di-impute
df = pd.concat([df_emp_length_null, df_emp_length_not_null]).sort_index()
```

3.3 Transformation

Merubah data menjadi numerical untuk pengolahan lebih lanjut.

- **Ordinal data**: grade & emp_length dapat diubah menjadi numerical.
- **Nominal data**: loan_status, home_ownership, term dapat diubah menjadi boolean numerical (1,0).

In [46]:
```
## Grade A-G -> 1-7, Employement Length 1-10 -> 1-10
replacing_dict = {
"emp_length": {
"10+ years": 10,
"9 years": 9,
"8 years": 8,
"7 years": 7,
"6 years": 6,
"5 years": 5,
"4 years": 4,
"3 years": 3,
"2 years": 2,
"1 year": 1,
"< 1 year": 0,
},
"grade":{
"A": 1,
"B": 2,
"C": 3,
"D": 4,
"E": 5,
"F": 6,
"G": 7
}
}
```

In [47]:
```
## Operasi replace ordinal -> numerical
df = df.replace(replacing_dict)
```

In [48]:
```
## Data dengan tipe ordinal telah diubah menjadi numerical
df.head()
```

Out[48]:

	loan_status	loan_amnt	int_rate	grade	emp_length	home_ownership	annual_inc	term
0	Current	2500	13.56	3	10	RENT	55000	36 months
1	Current	30000	18.94	4	10	MORTGAGE	90000	60 months
2	Current	5000	17.97	4	6	MORTGAGE	59280	36 months
3	Current	4000	18.94	4	10	MORTGAGE	92000	36 months
4	Current	30000	16.14	3	10	MORTGAGE	57250	60 months

In [49]:
```
## Nominal columns -> numerical columns
nominal_columns = ["home_ownership", "loan_status", "term"]
## Membuat dummy df untuk nominal columns
dummy_df = pd.get_dummies(df[nominal_columns])
```

In [50]:
```
## Data nominal telah diubah menjadi numerical
dummy_df.head()
```

Out[50]:

	home_ownership_ANY	home_ownership_MORTGAGE	home_ownership_OWN	home_ownership_RENT	loan_status_Charged Off	loan_
0	0	0	0	1	0	
1	0	1	0	0	0	
2	0	1	0	0	0	
3	0	1	0	0	0	
4	0	1	0	0	0	

In [51]:
```
## Update df dengan data terbaru, yaitu data yang hanya mengandung numerical features
df = pd.concat([df.drop(nominal_columns, axis=1), dummy_df], axis=1)
```

3.4 Outlier Treatment

Aturan *outlier treatment* yang digunakan adalah membuat threshold menggunakan:

- Q1 - 1.5 IQR, Q3 + 1.5 IQR
- Mean +- 2 SD

In [52]:
```
## Import library untuk visualisasi
import seaborn as sns
```

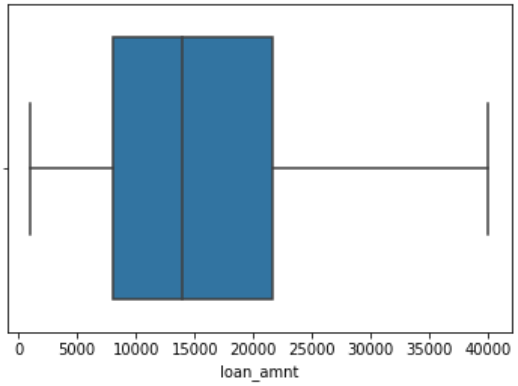
In [53]:
```
## Hasil tabel yang telah di lakukan transformation
df.head()
```

Out[53]:

	loan_amnt	int_rate	grade	emp_length	annual_inc	home_ownership_ANY	home_ownership_MORTGAGE	home_ownership_OWN
0	2500	13.56	3	10	55000	0	0	0
1	30000	18.94	4	10	90000	0	1	0
2	5000	17.97	4	6	59280	0	1	0
3	4000	18.94	4	10	92000	0	1	0
4	30000	16.14	3	10	57250	0	1	0

In [54]:
```
## Check outlier loan_amnt
sns.boxplot(df['loan_amnt'])
```

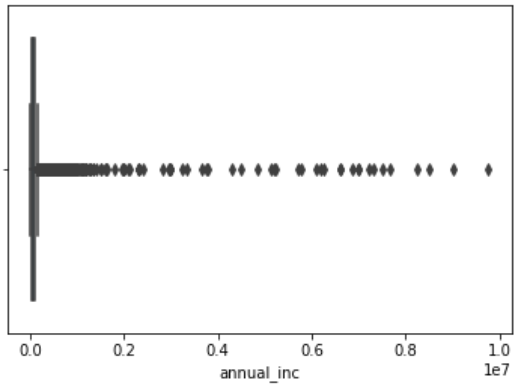
Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x10349330>
```



Kolom loan_amnt tidak memiliki outlier, tidak perlu dilakukan treatment

In [55]:
```
## Check outlier annual_inc
sns.boxplot(df['annual_inc'])
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x113bb230>
```



Kolom annual_inc memiliki banyak sekali outliers, outliers treatment perlu dilakukan.

In [56]:
```
## Mengambil Q1, Q3, dan IQR kolom annual_inc
q1_annual_inc = df['annual_inc'].quantile(0.25)
q3_annual_inc = df['annual_inc'].quantile(0.75)
iqr_annual_inc = q3_annual_inc - q1_annual_inc
```

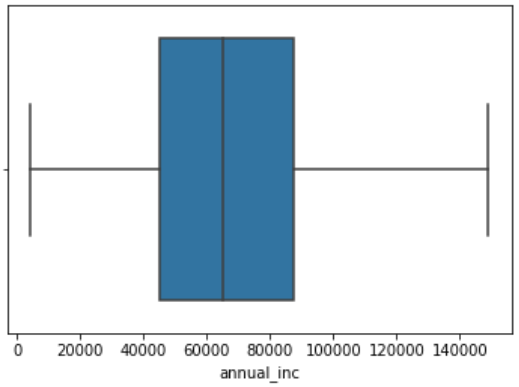
In [57]:
```
lower_bound_annual_inc = iqr_annual_inc - q1_annual_inc
upper_bound_annual_inc = iqr_annual_inc + q3_annual_inc
```

In [58]:
```
## Check shape sebelum diremove outliers
df.shape
```

Out[58]: (148745, 17)

In [59]:
```
## Remove outliers menggunakan > 0.95 & < 0.05 karena banyaknya outliers yang terdeteksi
df = df[(df['annual_inc'] > lower_bound_annual_inc) & (df['annual_inc'] < upper_bound_annual_inc)]
sns.boxplot(df['annual_inc'])
```

Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x114b6390>



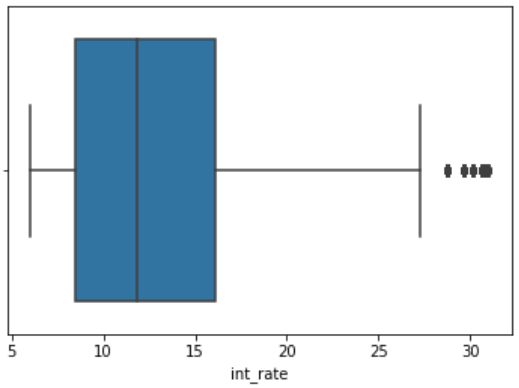
In [60]:
```
## Check shape setelah diremove outliers
df.shape
```

Out[60]: (135433, 17)

Outlers pada kolom annual_inc telah dihilangkan, dapat dilihat distribusi boxplot di atas.

In [61]:
```
## Check outlier int_rate
sns.boxplot(df['int_rate'])
```

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0xb88a7b0>



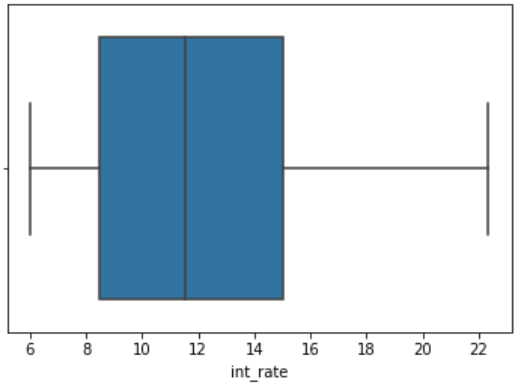
In [62]:
```
## Check shape sebelum diremove outliers
df.shape
```

Out[62]: (135433, 17)

In [63]:
```
## Remove outliers pada kolom int_rate
mean_interest_rate = df['int_rate'].mean()
std_interest_rate = df['int_rate'].std()
upper_interest_rate = mean_interest_rate + (2*std_interest_rate)
lower_interest_rate = mean_interest_rate - (2*std_interest_rate)
```

In [64]:
```
## Outliers telah diremove
df = df[(df['int_rate'] < upper_interest_rate) & (df['int_rate'] > lower_interest_rate)]
sns.boxplot(df['int_rate'])
```

Out[64]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb7e7030>
```



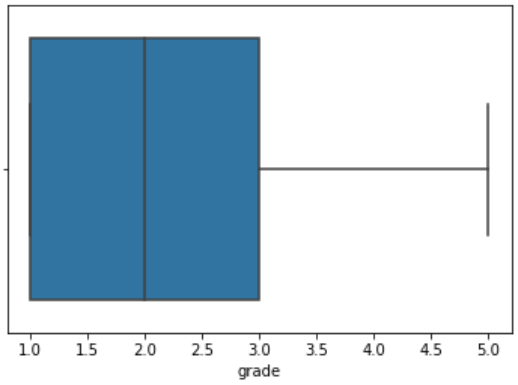
In [65]:
```
## Check shape setelah diremove outliers
df.shape
```

Out[65]:
```
(128548, 17)
```

Outlers pada kolom int_rate telah dihilangkan, dapat dilihat distribusi boxplot di atas.

In [66]:
```
## Check outlier grade
sns.boxplot(df['grade'])
```

Out[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb820590>
```

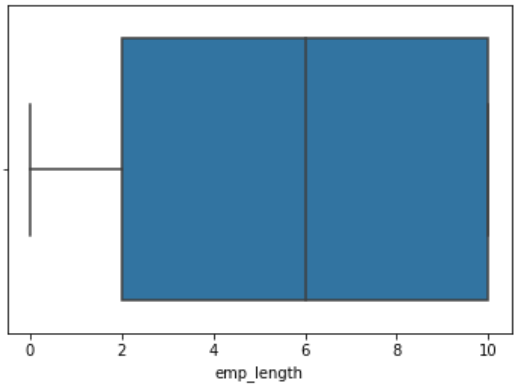


In [67]:
```
df['grade'].describe()
```

Out[67]:
```
count    128548.000000
mean          2.240012
std           1.040200
min           1.000000
25%           1.000000
50%           2.000000
75%           3.000000
max           5.000000
Name: grade, dtype: float64
```

In [68]:
```
## Check outlier emp_length
sns.boxplot(df['emp_length'])
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb7db4b0>
```



In [69]:
```
df['emp_length'].describe()
```

Out[69]:
```
count    128548.000000
mean          5.933029
std           3.883395
min           0.000000
25%           2.000000
50%           6.000000
75%          10.000000
max          10.000000
Name: emp_length, dtype: float64
```

Kolom **grade** dan **emp_length** tidak memiliki outliers, hanya saja distribusinya *skewed* karena banyaknya data yang terdapat pada salah satu bagian (negatif/positif) dari distribusi.

4. Export Result

In [70]:
```
## Check final shape
df.shape
```

Out[70]:
```
(128548, 17)
```

In [71]:
```
df.to_csv("data-t1-result.csv", index=False)
```