0. Import Library

In [5]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import sklearn as
import datetime
```

Import Data

In [6]:

```
loan = pd.read_csv('kiva_loans.csv')
loan.head()
```

Out[6]:

	id	funded_amount	loan_amount	activity	sector	use	country_code	country	region	currency	part
0	653051	300.0	300.0	Fruits & Vegetables	Food	To buy seasonal, fresh fruits to sell.	PK	Pakistan	Lahore	PKR	
1	653053	575.0	575.0	Rickshaw	Transportation	to repair and maintain the auto rickshaw used ...	PK	Pakistan	Lahore	PKR	
2	653068	150.0	150.0	Transportation	Transportation	To repair their old cycle-van and buy another ...	IN	India	Maynaguri	INR	
						to purchase					

In [7]:

```
geolocation = pd.read_csv('kiva_mpi_region_locations.csv')
geolocation.head()
```

Out[7]:

	LocationName	ISO	country	region	world_region	MPI	geo	lat	lon
0	Badakhshan, Afghanistan	AFG	Afghanistan	Badakhshan	South Asia	0.387	(36.7347725, 70.81199529999999)	36.734772	70.811995
1	Badghis, Afghanistan	AFG	Afghanistan	Badghis	South Asia	0.466	(35.1671339, 63.7695384)	35.167134	63.769538
2	Baghlan, Afghanistan	AFG	Afghanistan	Baghlan	South Asia	0.300	(35.8042947, 69.2877535)	35.804295	69.287754
3	Balkh, Afghanistan	AFG	Afghanistan	Balkh	South Asia	0.301	(36.7550603, 66.8975372)	36.755060	66.897537
4	Bamyan, Afghanistan	AFG	Afghanistan	Bamyan	South Asia	0.325	(34.8100067, 67.8212104)	34.810007	67.821210

In [8]:

```
theme = pd.read_csv('loan_theme_ids.csv')
theme.head()
```

Out[8]:

	id	Loan Theme ID	Loan Theme Type	Partner ID
0	638631	a1050000000skGl	General	151.0
1	640322	a1050000000skGl	General	151.0
2	641006	a1050000002X1ij	Higher Education	160.0
3	641019	a1050000002X1ij	Higher Education	160.0
4	641594	a1050000002VbsW	Subsistence Agriculture	336.0

In [9]:
```
theme_location = pd.read_csv('loan_themes_by_region.csv')
theme_location.head()
```

Out[9]:

	Partner ID	Field Partner Name	sector	Loan Theme ID	Loan Theme Type	country	forkiva	region	geocode_old	ISO	...	amount	L
0	9	KREDIT Microfinance Institution	General Financial Inclusion	a1050000000slfi	Higher Education	Cambodia	No	Banteay Meanchey	(13.75, 103.0)	KHM	...	450	
1	9	KREDIT Microfinance Institution	General Financial Inclusion	a10500000068jPe	Vulnerable Populations	Cambodia	No	Battambang Province	NaN	KHM	...	20275	
2	9	KREDIT Microfinance Institution	General Financial Inclusion	a1050000000slfi	Higher Education	Cambodia	No	Battambang Province	NaN	KHM	...	9150	
3	9	KREDIT Microfinance Institution	General Financial Inclusion	a10500000068jPe	Vulnerable Populations	Cambodia	No	Kampong Cham Province	(12.0, 105.5)	KHM	...	604950	

In [10]:
```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 671205 entries, 0 to 671204
Data columns (total 20 columns):
id                    671205 non-null int64
funded_amount         671205 non-null float64
loan_amount           671205 non-null float64
activity              671205 non-null object
sector                671205 non-null object
use                   666973 non-null object
country_code          671197 non-null object
country               671205 non-null object
region                614405 non-null object
currency              671205 non-null object
partner_id            657698 non-null float64
posted_time           671205 non-null object
disbursed_time        668809 non-null object
funded_time           622874 non-null object
term_in_months        671205 non-null float64
lender_count          671205 non-null int64
tags                  499789 non-null object
borrower_genders      666984 non-null object
repayment_interval    671205 non-null object
date                  671205 non-null object
dtypes: float64(4), int64(2), object(14)
memory usage: 102.4+ MB
```

In [11]:
```
theme.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 779092 entries, 0 to 779091
Data columns (total 4 columns):
id                   779092 non-null int64
Loan Theme ID        764279 non-null object
Loan Theme Type      764279 non-null object
Partner ID           764279 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 23.8+ MB
```

In [12]:
```
geolocation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2772 entries, 0 to 2771
Data columns (total 9 columns):
LocationName    984 non-null object
ISO             1008 non-null object
country         1008 non-null object
region          984 non-null object
world_region    1008 non-null object
MPI             984 non-null float64
geo             2772 non-null object
lat             892 non-null float64
lon             892 non-null float64
dtypes: float64(3), object(6)
memory usage: 195.0+ KB
```

In [13]:
```
theme_location.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15736 entries, 0 to 15735
Data columns (total 21 columns):
Partner ID            15736 non-null int64
Field Partner Name    15736 non-null object
sector                15736 non-null object
Loan Theme ID         15736 non-null object
Loan Theme Type       15736 non-null object
country               15736 non-null object
forkiva               15736 non-null object
region                15736 non-null object
geocode_old           1200 non-null object
ISO                   15722 non-null object
number                15736 non-null int64
amount                15736 non-null int64
LocationName          15736 non-null object
geocode               13662 non-null object
names                 13661 non-null object
geo                   15736 non-null object
lat                   13662 non-null float64
lon                   13662 non-null float64
mpi_region            15722 non-null object
mpi_geo               9671 non-null object
rural_pct             14344 non-null float64
dtypes: float64(3), int64(3), object(15)
memory usage: 2.5+ MB
```

2. Data Relation Analysis

2.1 Column View

- Theme CSV:

  Columns

  id Unique ID for loan (Loan ID)

  A Loan Theme ID ID for Loan Theme

  A Loan Theme Type Category name of type of loan

  \# Partner ID

- Geolocation CSV

Columns

- A LocationName region, country
- A ISO some sort of unique abbreviation for country
- A country country
- A region region with in country
- A world_region parts of the world
- # MPI multidimensional poverty index
- A geo (latitude, longitude)
- lat latitude
- lon longitude

- Loan CSV:

Columns

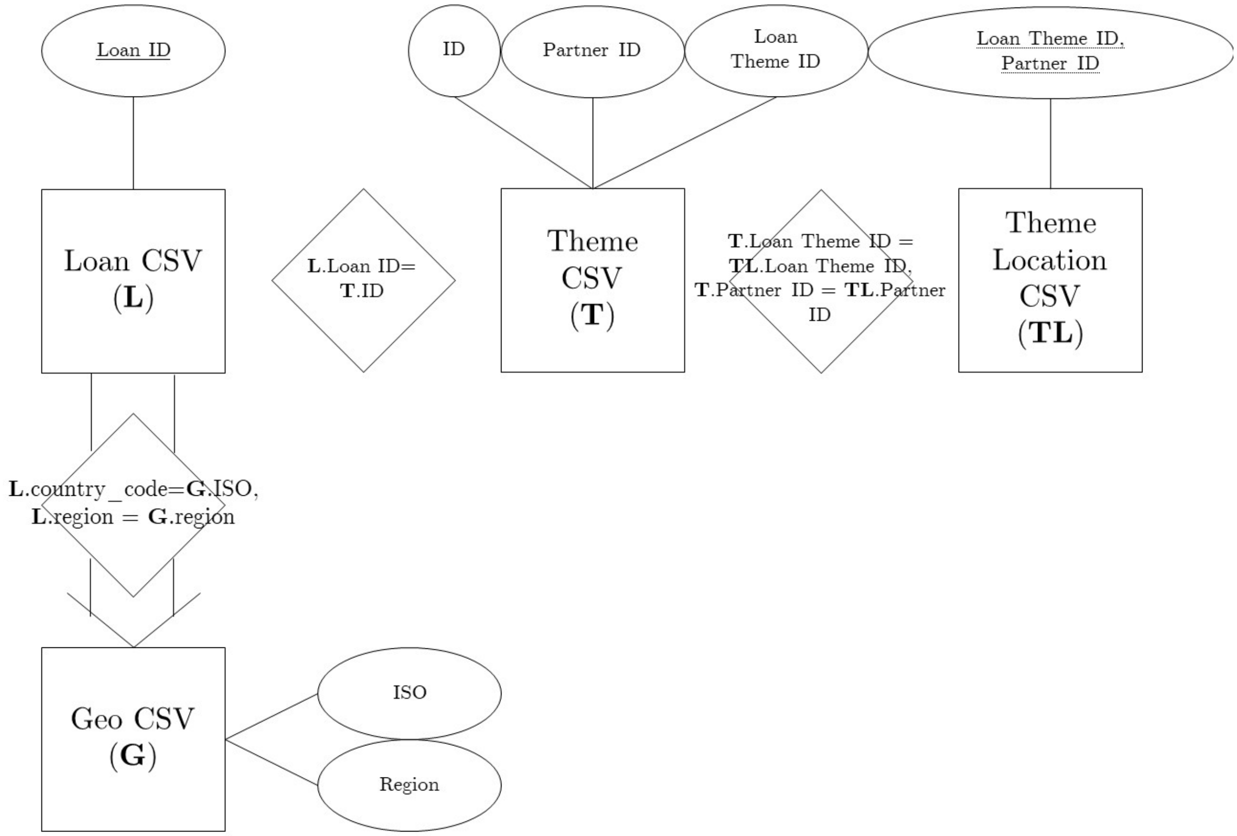
- id Unique ID for loan
- # funded_amount The amount disbursed by Kiva to the field agent(USD)
- # loan_amount The amount disbursed by the field agent to the borrower(USD)
- A activity More granular category
- A sector High level category
- A use Exact usage of loan amount
- country_code ISO country code of country in which loan was disbursed
- A country Full country name of country in which loan was disbursed
- A region Full region name within the country
- A currency The currency in which the loan was disbursed
- # partner_id ID of partner organization
- posted_time The time at which the loan is posted on Kiva by the field agent
- disbursed_time The time at which the loan is disbursed by the field agent to the borrower
- funded_time The time at which the loan posted to Kiva gets funded by lenders completely
- # term_in_months The duration for which the loan was disbursed in months
- # lender_count The total number of lenders that contributed to this loan
- A tags
- A borrower_genders Comma separated M,F letters, where each instance represents a single male/female in the group
- A repayment_interval
- date

- Theme Location

Columns

- Partner ID
- Field Partner Name
- sector
- Loan Theme ID
- Loan Theme Type
- country
- forkiva
- region
- geocode_old
- ISO
- number
- amount
- LocationName
- geocode
- names
- geo
- lat
- lon
- mpi_region
- mpi_geo
- rural_pct

2.2 General Relationship



The ER diagram in above shows the interaction between the CSV files. Loan is associated with Theme CSV, while Theme CSV associates with Theme Location CSV, giving more details about the loan. Geo CSV indicates the exact location of the loan. However, all CSV such as Theme CSV may not contains all Loan CSV loan ids, while same applies to the relationship between Theme CSV and Theme Location CSV, and Geolocation CSV and Loan CSV. Some of the values are missing, such as Theme CSV Theme ID are not even found in the Theme Location CSV, and Loan country_code not found in Geolocation CSV. Therefore, it is suggested that only a small subset of Loan contains the complete information of Theme and Partner, while all loan has access to the Geolocation.

2.3 Method to find the records between CSV

In [14]:
```
loan.loc[loan['id'] == 657249]
```

Out[14]:

	id	funded_amount	loan_amount	activity	sector	use	country_code	country	region	currency	partner_i
4132	657249	1450.0	1450.0	Agriculture	Agriculture	to buy agricultural supplies for growing certi...	HN	Honduras	La Esperanza, Intibuca	HNL	269.

In [15]:
```
theme.loc[theme['id'] == 657249]
```

Out[15]:

	id	Loan Theme ID	Loan Theme Type	Partner ID
4582	657249	a1050000000tsXd	Agriculture	269.0

In [16]:
```
theme_location.loc[(theme_location['Partner ID'] == 269) & (theme_location['Loan Theme ID'] == 'a1050000000ts
```

Out[16]:

Partner ID	Field Partner Name	sector	Loan Theme ID	Loan Theme Type	country	forkiva	region	geocode_old	ISO	...	amount	LocationName	geocode	names

0 rows × 21 columns

In [17]:
```
geolocation.loc[(geolocation['ISO'] == 'HN') & (geolocation['region'] == 'La Esperanza, Intibuca')]
```

Out[17]:

LocationName	ISO	country	region	world_region	MPI	geo	lat	lon

3. Data Preprocessing

As for the prelimary analysis, only the easy, simple conversion is focused.

3.1 Filter out incomplete data

In here, we extract a set of data that is has complete information as mentioned in 2.3. They must contains:

- complete Theme ID
- complete Partner Information
- complete Geolocation

In [18]:
```
df1 = pd.merge(loan,geolocation,how='inner')
```

In [19]:
```
df1.head()
```

Out[19]:

	id	funded_amount	loan_amount	activity	sector	use	country_code	country	region	currency	...	borrowe
0	653359	600.0	600.0	Machinery Rental	Services	to invest in working capital and to maintain g...	NI	Nicaragua	Leon	NIO	...	
1	653373	1000.0	1000.0	Grocery Store	Food	to invest in working capital and to provide hi...	NI	Nicaragua	Leon	NIO	...	
2	653364	250.0	250.0	Fuel/Firewood	Retail	to buy firewood to offer to her customers	NI	Nicaragua	Leon	NIO	...	

In [20]:
```
df1.columns
```

Out[20]:
```
Index(['id', 'funded_amount', 'loan_amount', 'activity', 'sector', 'use',
       'country_code', 'country', 'region', 'currency', 'partner_id',
       'posted_time', 'disbursed_time', 'funded_time', 'term_in_months',
       'lender_count', 'tags', 'borrower_genders', 'repayment_interval',
       'date', 'LocationName', 'ISO', 'world_region', 'MPI', 'geo', 'lat',
       'lon'],
      dtype='object')
```

In [21]:
```
df1 = pd.merge(df1,theme,how='inner')
```

In [22]:
```
df1.columns
```

Out[22]:
```
Index(['id', 'funded_amount', 'loan_amount', 'activity', 'sector', 'use',
       'country_code', 'country', 'region', 'currency', 'partner_id',
       'posted_time', 'disbursed_time', 'funded_time', 'term_in_months',
       'lender_count', 'tags', 'borrower_genders', 'repayment_interval',
       'date', 'LocationName', 'ISO', 'world_region', 'MPI', 'geo', 'lat',
       'lon', 'Loan Theme ID', 'Loan Theme Type', 'Partner ID'],
      dtype='object')
```

In [23]:
```
df1.head()
```

Out[23]:

	id	funded_amount	loan_amount	activity	sector	use	country_code	country	region	currency	...	LocationNa
0	653359	600.0	600.0	Machinery Rental	Services	to invest in working capital and to maintain g...	NI	Nicaragua	Leon	NIO	...	Le Nicara
1	653373	1000.0	1000.0	Grocery Store	Food	to invest in working capital and to provide hi...	NI	Nicaragua	Leon	NIO	...	Le Nicara
2	653364	250.0	250.0	Fuel/Firewood	Retail	to buy firewood to offer to her customers	NI	Nicaragua	Leon	NIO	...	Le Nicara
3	653367	175.0	175.0	Food Market	Food	to invest in working capital and stock up her ...	NI	Nicaragua	Leon	NIO	...	Le Nicara
4	653396	800.0	800.0	Clothing Sales	Clothing	to invest in working capital to stock their bu...	NI	Nicaragua	Leon	NIO	...	Le Nicara

5 rows × 30 columns

In [24]:
```
df1 = pd.merge(df1,theme_location,how='inner')
```

In [25]:
```
df1.columns
```

Out[25]:
```
Index(['id', 'funded_amount', 'loan_amount', 'activity', 'sector', 'use',
       'country_code', 'country', 'region', 'currency', 'partner_id',
       'posted_time', 'disbursed_time', 'funded_time', 'term_in_months',
       'lender_count', 'tags', 'borrower_genders', 'repayment_interval',
       'date', 'LocationName', 'ISO', 'world_region', 'MPI', 'geo', 'lat',
       'lon', 'Loan Theme ID', 'Loan Theme Type', 'Partner ID',
       'Field Partner Name', 'forkiva', 'geocode_old', 'number', 'amount',
       'geocode', 'names', 'mpi_region', 'mpi_geo', 'rural_pct'],
      dtype='object')
```

In [26]:
```
df1.head()
```

Out[26]:

	id	funded_amount	loan_amount	activity	sector	use	country_code	country	region	currency	...	Field Partner Name	f
0	773240	725.0	725.0	Higher education costs	Education	To pay for university tuition fees and cost of...	KH	Cambodia	Phnom Penh	USD	...	Wedu	
1	773242	1000.0	1000.0	Higher education costs	Education	To pay for university fees and cost of living ...	KH	Cambodia	Phnom Penh	USD	...	Wedu	
2	1035120	2000.0	2000.0	Higher education	Education	To pay for university	KH	Cambodia	Phnom	USD	...	Wedu	

3.2 One-hot encoding

In order to get rid of the non-numeric features, one-hot encoding is necessary. However, there are too many options, resulting a sparse matrix. In current stage, only selected features are processed. They are:

- `repayment_interval`
- `ISO`
- `Loan Theme Type` (*note that loan theme type and loan theme id are different*)
- `Gender` (*In case of more than one applicant, we choose mix*)
- `Number of applicant` (*extracted by counting the number of applicats in `gender`*)
- `Currency`

For the other attributes, they will not be preserved otherwise selected.

- `id`
- `funded_amount`
- `loan_amount`
- `currency`
- `term_in_months`
- `lender_count`
- `borrower_genders`
- `repayment_interval`
- `ISO`
- `Loan Theme Type`
- `number`
- `amount`

These attributes will add into consideration in future:

- `posted_time`
- `disbursed_time`
- `funded_time`
- `date`

In [27]:
```
df2 = df1[['id','funded_amount','loan_amount','currency','term_in_months','lender_count','borrower_genders','
df2.columns
```

Out[27]:
```
Index(['id', 'funded_amount', 'loan_amount', 'currency', 'term_in_months',
       'lender_count', 'borrower_genders', 'repayment_interval', 'ISO',
       'Loan Theme Type', 'number', 'amount'],
      dtype='object')
```

In [28]:
```
df2.head()
```

Out[28]:

	id	funded_amount	loan_amount	currency	term_in_months	lender_count	borrower_genders	repayment_interval	ISO	L The T
0	773240	725.0	725.0	USD	38.0	29	female	bullet	KHM	Hui Ca Contr
1	773242	1000.0	1000.0	USD	101.0	21	female	bullet	KHM	Hui Ca Contr
2	1035120	2000.0	2000.0	USD	120.0	46	female	bullet	KHM	Hui Ca Contr
3	1151588	1600.0	1600.0	USD	72.0	55	female	bullet	KHM	Hui Ca Contr
4	1210828	3600.0	3600.0	USD	110.0	122	female	bullet	KHM	Hui Ca Contr

In [29]:
```
# print(df2.iloc[0]['funded_time'][:-6])
# print(datetime.datetime.strptime(df2.iloc[0]['funded_time'][:-6], '%Y-%m-%d %H:%M:%S'))
```

In [30]:
```
choices = set(df2['repayment_interval'])
df3 = df2
for each in choices:
    dummy = pd.DataFrame({
        each:[1 if row['repayment_interval']==each else 0 for index, row in df3.iterrows()]
    })
    df3 = df3.join(dummy)
df3 = df3.drop('repayment_interval', axis=1)
```

In [31]:
```
df3.head()
```

Out[31]:

	id	funded_amount	loan_amount	currency	term_in_months	lender_count	borrower_genders	ISO	Loan Theme Type	number	amou
0	773240	725.0	725.0	USD	38.0	29	female	KHM	Human Capital Contracts	3	72(
1	773242	1000.0	1000.0	USD	101.0	21	female	KHM	Human Capital Contracts	3	72(
2	1035120	2000.0	2000.0	USD	120.0	46	female	KHM	Human Capital Contracts	3	72(
3	1151588	1600.0	1600.0	USD	72.0	55	female	KHM	Human Capital Contracts	3	72(
4	1210828	3600.0	3600.0	USD	110.0	122	female	KHM	Human Capital Contracts	3	72(

In [32]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6718 entries, 0 to 6717
Data columns (total 14 columns):
id                  6718 non-null int64
funded_amount       6718 non-null float64
loan_amount         6718 non-null float64
currency            6718 non-null object
term_in_months      6718 non-null float64
lender_count        6718 non-null int64
borrower_genders    6718 non-null object
ISO                 6718 non-null object
Loan Theme Type     6718 non-null object
number              6718 non-null int64
amount              6718 non-null int64
monthly             6718 non-null int64
bullet              6718 non-null int64
irregular           6718 non-null int64
dtypes: float64(3), int64(7), object(4)
memory usage: 1.1+ MB
```

In [33]:
```
choices = set(df3['currency'])
print(choices)
df4 = df3
for each in choices:
    dummy = pd.DataFrame({
        each:[1 if row['currency']==each else 0 for index, row in df4.iterrows()]
    })
    df4 = df4.join(dummy)
df4 = df4.drop('currency', axis=1)
```

```
{'KES', 'USD', 'PEN', 'NIO', 'ZMW', 'NGN', 'IDR'}
```

In [34]:
```
df4.tail()
```

Out[34]:

	id	funded_amount	loan_amount	term_in_months	lender_count	borrower_genders	ISO	Loan Theme Type	number	amount	mc
6713	1217745	1925.0	1925.0	8.0	66	male, male, male, male	NIC	Agriculture	309	90950	
6714	1218818	1225.0	1225.0	10.0	43	male, male, female	NIC	Agriculture	309	90950	
6715	1252366	1375.0	1375.0	10.0	31	female, female, female	NIC	Agriculture	309	90950	
6716	1324972	875.0	875.0	9.0	33	female, male, female, female	NIC	Agriculture	309	90950	
6717	1300428	5050.0	5050.0	50.0	147	female	PER	Higher Education	1	5050	

In [35]:
```
df4.columns
```

Out[35]:
```
Index(['id', 'funded_amount', 'loan_amount', 'term_in_months', 'lender_count',
       'borrower_genders', 'ISO', 'Loan Theme Type', 'number', 'amount',
       'monthly', 'bullet', 'irregular', 'KES', 'USD', 'PEN', 'NIO', 'ZMW',
       'NGN', 'IDR'],
      dtype='object')
```

In [36]:
```
choices = set(df4['Loan Theme Type'])
print(choices)
df5 = df4
for each in choices:
    dummy = pd.DataFrame({
        each:[1 if row['Loan Theme Type']==each else 0 for index, row in df5.iterrows()]
    })
    df5 = df5.join(dummy)
df5 = df5.drop('Loan Theme Type', axis=1)
```

```
{'Bridge/Income Smoothing', 'Full Tuition', 'Cacao Field Renewal', 'Agriculture', 'Value Chain', 'Education
Technology', 'Human Capital Contracts', 'Field Renewal', 'Higher Education', 'Coffee Production', 'Cacao Pro
duction', 'Agricultural Infrastructure'}
```

In [37]:
```
df5.columns
```

Out[37]:
```
Index(['id', 'funded_amount', 'loan_amount', 'term_in_months', 'lender_count',
       'borrower_genders', 'ISO', 'number', 'amount', 'monthly', 'bullet',
       'irregular', 'KES', 'USD', 'PEN', 'NIO', 'ZMW', 'NGN', 'IDR',
       'Bridge/Income Smoothing', 'Full Tuition', 'Cacao Field Renewal',
       'Agriculture', 'Value Chain', 'Education Technology',
       'Human Capital Contracts', 'Field Renewal', 'Higher Education',
       'Coffee Production', 'Cacao Production', 'Agricultural Infrastructure'],
      dtype='object')
```

In [38]:
```
df5.head()
```

Out[38]:

	id	funded_amount	loan_amount	term_in_months	lender_count	borrower_genders	ISO	number	amount	monthly	...	C Ren
0	773240	725.0	725.0	38.0	29	female	KHM	3	7200	0	...	
1	773242	1000.0	1000.0	101.0	21	female	KHM	3	7200	0	...	
2	1035120	2000.0	2000.0	120.0	46	female	KHM	3	7200	0	...	
3	1151588	1600.0	1600.0	72.0	55	female	KHM	3	7200	0	...	
4	1210828	3600.0	3600.0	110.0	122	female	KHM	3	7200	0	...	

5 rows × 31 columns

In [39]:
```
choices = set(df5['ISO'])
print(choices)
df6 = df5
for each in choices:
    dummy = pd.DataFrame({
        each:[1 if row['ISO']==each else 0 for index, row in df6.iterrows()]
    })
    df6 = df6.join(dummy)
df6 = df6.drop('ISO', axis=1)
```

```
{'KEN', 'IDN', 'PER', 'ZMB', 'NGA', 'KHM', 'NIC'}
```

In [40]:
```
df6.columns
```

Out[40]:
```
Index(['id', 'funded_amount', 'loan_amount', 'term_in_months', 'lender_count',
       'borrower_genders', 'number', 'amount', 'monthly', 'bullet',
       'irregular', 'KES', 'USD', 'PEN', 'NIO', 'ZMW', 'NGN', 'IDR',
       'Bridge/Income Smoothing', 'Full Tuition', 'Cacao Field Renewal',
       'Agriculture', 'Value Chain', 'Education Technology',
       'Human Capital Contracts', 'Field Renewal', 'Higher Education',
       'Coffee Production', 'Cacao Production', 'Agricultural Infrastructure',
       'KEN', 'IDN', 'PER', 'ZMB', 'NGA', 'KHM', 'NIC'],
      dtype='object')
```

In [41]:
```
df6.tail()
```

Out[41]:

	id	funded_amount	loan_amount	term_in_months	lender_count	borrower_genders	number	amount	monthly	bullet	...
6713	1217745	1925.0	1925.0	8.0	66	male, male, male, male	309	90950	0	1	...
6714	1218818	1225.0	1225.0	10.0	43	male, male, female	309	90950	0	1	...
6715	1252366	1375.0	1375.0	10.0	31	female, female, female	309	90950	0	1	...
6716	1324972	875.0	875.0	9.0	33	female, male, female, female	309	90950	0	1	...
6717	1300428	5050.0	5050.0	50.0	147	female	1	5050	0	1	...

5 rows × 37 columns

In [42]:
```
string = df6.iloc[6716]['borrower_genders']
gender_list = string.split(', ')
print(gender_list)
print(len(gender_list))
print('female' in gender_list)
print('female' in string)
```

```
['female', 'male', 'female', 'female']
4
True
True
```

In [43]:
```
string = df6.iloc[6717]['borrower_genders']
gender_list = string.split(', ')
print(gender_list)
print(len(gender_list))
print('female' in gender_list)
print('female' in string)
print('male' in string)
print('male' in gender_list)
```

```
['female']
1
True
True
True
False
```

In [87]:
```
df7 = df6
dummy = pd.DataFrame({
    'no_of_borrowers':[len(row['borrower_genders'].split(', ')) for index, row in df7.iterrows()],
    'has_female':[1 if 'female' in row['borrower_genders'].split(', ') else 0 for index, row in df7.iterrows(
    'has_male':[1 if 'male' in row['borrower_genders'].split(', ') else 0 for index, row in df7.iterrows()],
})
df7 = df7.join(dummy)
df7 = df7.drop('borrower_genders', axis=1)
```

In [88]:
```
df7.tail()
```

Out[88]:

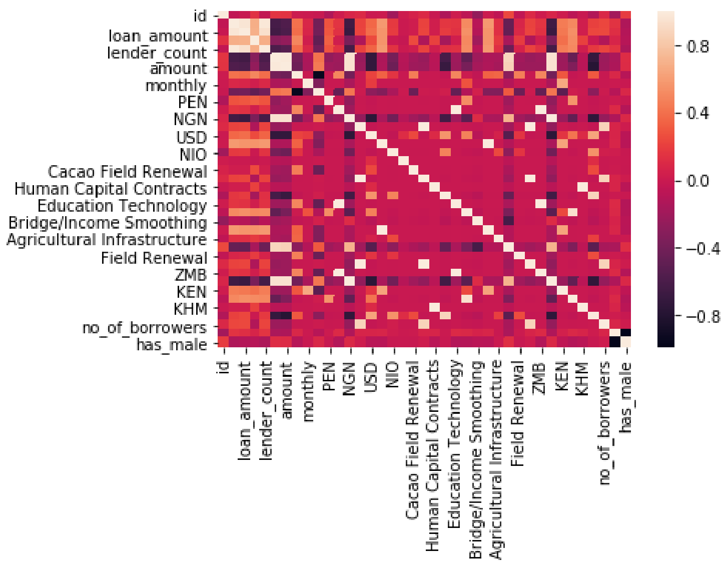
	id	funded_amount	loan_amount	term_in_months	lender_count	number	amount	monthly	bullet	irregular	...	KEN	IDI
6713	1217745	1925.0	1925.0	8.0	66	309	90950	0	1	0	...	0	(
6714	1218818	1225.0	1225.0	10.0	43	309	90950	0	1	0	...	0	(
6715	1252366	1375.0	1375.0	10.0	31	309	90950	0	1	0	...	0	(
6716	1324972	875.0	875.0	9.0	33	309	90950	0	1	0	...	0	(
6717	1300428	5050.0	5050.0	50.0	147	1	5050	0	1	0	...	0	(

5 rows × 39 columns

3.3 Short Review

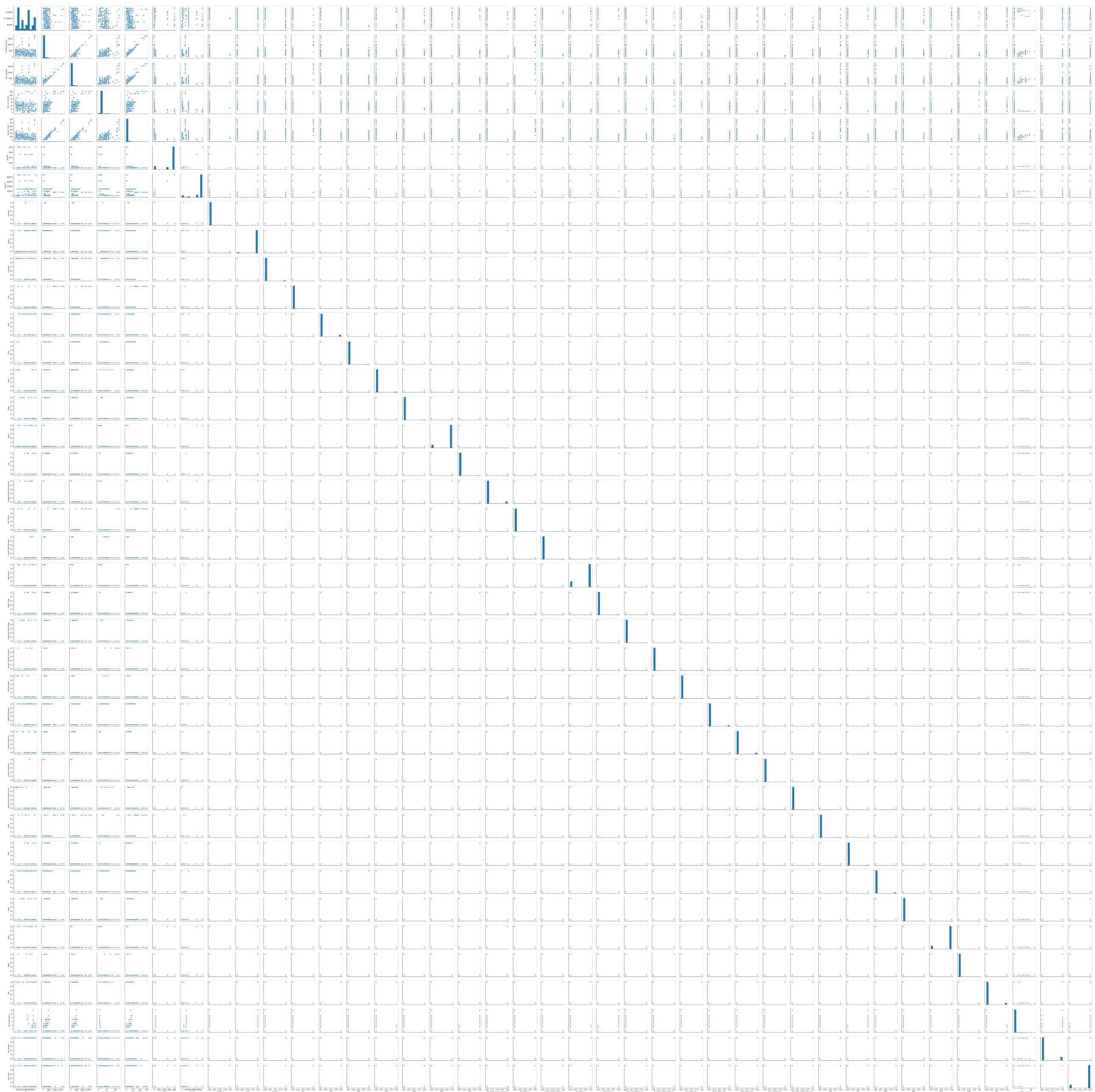
In [47]:
```
sns.heatmap(df7.corr())
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1f130daabe0>



In [46]:
```
sns.pairplot(df7)
```

Out[46]: <seaborn.axisgrid.PairGrid at 0x20a062d0a90>



4. Classification

In here, this is just a simple exploration of dataset using logistic regression. A simple standarization is preprocessed before test.

In [132]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix
```

In [116]:
```
df8 = df7
df8[:] = scaler.fit_transform(df7[:])
df8.head()
```

Out[116]:

	id	funded_amount	loan_amount	term_in_months	lender_count	number	amount	monthly	bullet	irregular	...	KEN	IDN
0	0.143791	0.041429	0.040057	0.300000	0.048013	0.000495	0.005746	0.0	1.0	0.0	...	0.0	0.0
1	0.143794	0.057143	0.055794	0.825000	0.034768	0.000495	0.005746	0.0	1.0	0.0	...	0.0	0.0
2	0.540396	0.114286	0.113019	0.983333	0.076159	0.000495	0.005746	0.0	1.0	0.0	...	0.0	0.0
3	0.716781	0.091429	0.090129	0.583333	0.091060	0.000495	0.005746	0.0	1.0	0.0	...	0.0	0.0
4	0.806497	0.205714	0.204578	0.900000	0.201987	0.000495	0.005746	0.0	1.0	0.0	...	0.0	0.0

5 rows × 39 columns

In [117]:
```
df8.columns
```

Out[117]:
```
Index(['id', 'funded_amount', 'loan_amount', 'term_in_months', 'lender_count',
       'number', 'amount', 'monthly', 'bullet', 'irregular', 'KES', 'USD',
       'PEN', 'NIO', 'ZMW', 'NGN', 'IDR', 'Bridge/Income Smoothing',
       'Full Tuition', 'Cacao Field Renewal', 'Agriculture', 'Value Chain',
       'Education Technology', 'Human Capital Contracts', 'Field Renewal',
       'Higher Education', 'Coffee Production', 'Cacao Production',
       'Agricultural Infrastructure', 'KEN', 'IDN', 'PER', 'ZMB', 'NGA', 'KHM',
       'NIC', 'no_of_borrowers', 'has_female', 'has_male'],
      dtype='object')
```

4.1 Classification on `bullet`

In [138]:
```
y = df8['bullet']
X = df8.loc[:, (df7.columns != 'monthly') & (df8.columns != 'irregular') & (df8.columns != 'bullet')]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [139]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [140]:
```
X_train.columns
```

Out[140]:
```
Index(['id', 'funded_amount', 'loan_amount', 'term_in_months', 'lender_count',
       'number', 'amount', 'KES', 'USD', 'PEN', 'NIO', 'ZMW', 'NGN', 'IDR',
       'Bridge/Income Smoothing', 'Full Tuition', 'Cacao Field Renewal',
       'Agriculture', 'Value Chain', 'Education Technology',
       'Human Capital Contracts', 'Field Renewal', 'Higher Education',
       'Coffee Production', 'Cacao Production', 'Agricultural Infrastructure',
       'KEN', 'IDN', 'PER', 'ZMB', 'NGA', 'KHM', 'NIC', 'no_of_borrowers',
       'has_female', 'has_male'],
      dtype='object')
```

In [141]:
```
y_train.head()
```

Out[141]:
```
6074    1.0
1222    1.0
3815    1.0
4035    1.0
5605    1.0
Name: bullet, dtype: float64
```

In [142]:
```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
c:\program files\python37\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default sol
ver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[142]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [143]:
```
predictions = logmodel.predict(X_test)
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        52
         1.0       1.00      1.00      1.00      2165

   micro avg       1.00      1.00      1.00      2217
   macro avg       1.00      1.00      1.00      2217
weighted avg       1.00      1.00      1.00      2217
```

In [144]:
```
print(confusion_matrix(y_test, predictions))
```

```
[[  52    0]
 [   0 2165]]
```

4.2 Classification no `monthly`

In [145]:
```
y = df8['monthly']
X = df8.loc[:, (df8.columns != 'bullet') & (df8.columns != 'monthly') & (df8.columns != 'irregular')]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [146]:
```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
c:\program files\python37\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default sol
ver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[146]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [147]:
```
predictions = logmodel.predict(X_test)
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2209
         1.0       1.00      1.00      1.00         8

   micro avg       1.00      1.00      1.00      2217
   macro avg       1.00      1.00      1.00      2217
weighted avg       1.00      1.00      1.00      2217
```

In [148]:
```
print(confusion_matrix(y_test, predictions))
```

```
[[2209    0]
 [   0    8]]
```

4.3 Classification on `irregular`

In [149]:
```
y = df8['irregular']
X = df8.loc[:, (df8.columns != 'bullet') & (df8.columns != 'monthly') & (df8.columns != 'irregular')]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [150]:
```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
c:\program files\python37\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default sol
ver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[150]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [151]:
```
predictions = logmodel.predict(X_test)
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2173
         1.0       1.00      0.95      0.98        44

   micro avg       1.00      1.00      1.00      2217
   macro avg       1.00      0.98      0.99      2217
weighted avg       1.00      1.00      1.00      2217
```

In [152]:
```
print(confusion_matrix(y_test, predictions))
```

```
[[2173    0]
 [   2   42]]
```

4.4 Experimental Result

However, the result is excellent; most prediction achieves over 95% accuracy, in any indicators (F1, recall, support, etc.).

5. Future Work

In future, more work should be done on exploring the accuracy of the result. There are three key areas to work with:

- **Explore full dataset**

This notebook only explores the range of dataset which contains all the information. Outside the selected data, there are lots of cases that do not have complete information. To handle this, substitution is required such as replacing missing values with average or discarding the data. The lost of information also hinder the prediction result, as not all the features are available for each data.

- **Try different prediction model**

In this notebook, only logistic regression is used. Thus, a more complete view is to try SVM, Decision Tree and LDA. Similar procedure can be referenced in the above work.

- **Feature Extraction**

Even this notebook has selected a small range of dataset, the dataset is already a sparse matrix. Most of the cell values are 0. To have better prediction, feature extraction such as best subset selection or Lasso should be used.

In [ ]:

```

```