IST-652
Homework1: Structured Data

Data and source

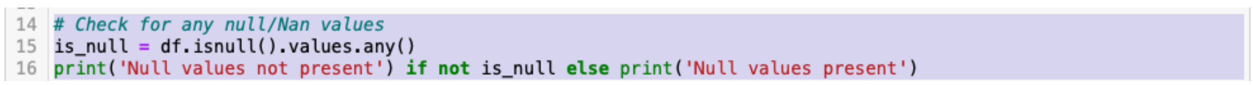
For the purpose of Homework-1 and to satisfy the requirement of structured-data, the following from the files section under IST-652 on the 2SU platform were used:

- Donor-Data.csv: comprising the donor data across various locations and donation amounts along-with other data
- data_dictionarydonors_data: map column-names in Donor-Data.csv to a definition/description of the field which helped understand the data better

Data exploration and clean-up

The first step is to familiarize with the dataset and dive into some data exploration and clean-up steps. For this, the csv was loaded into a *pandas* data-frame using a field separator or delimiter (in this case a comma). Subsequently, the dataset was examined using the following:

- **Check for null/NaN or missing values**
  It is a good practice to replace null/NaN or missing values with defaults so there are no exceptions when ingesting large datasets. For instance, it is a good practice to replace missing numerical cells with an average/mean. Below is an excerpt of the code:

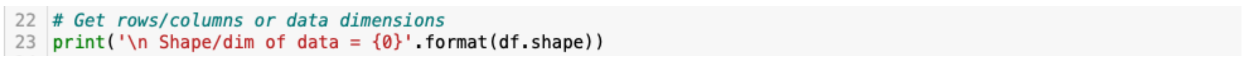
  ```
  # Check for any null/Nan values
  is_null = df.isnull().values.any()
  print('Null values not present') if not is_null else print('Null values present')
  ```

  In this case, it turned out that the dataset was clean.

  Output:
  ```
  Null values not present
  ```

- **View sample of dataset with head/tail**
  Using the *head* and *tail* functions examine the first and last few lines of the dataset.

- **Meet dataset requirements for Homework-1**
  shape function (which provides the rows, columns) helped verify the requirement of 500-4000 rows and 4-50 columns on the dataset:

  ```
  # Get rows/columns or data dimensions
  print('\n Shape/dim of data = {0}'.format(df.shape))
  ```

  Output
  ```
  Shape/dim of data = (3120, 24)
  ```

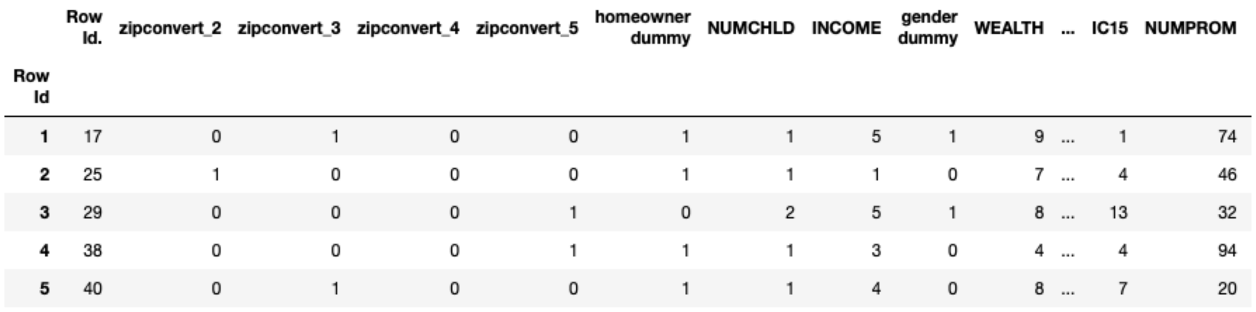
- **Inspect column-names**
  Map the column-names using the *column* property on the data-frame to the data_dictionarydonors_data file

  Output
  ```
  -- Column-names --
  ```

```
Index(['Row Id', 'Row Id.', 'zipconvert_2', 'zipconvert_3', 'zipconv
ert_4','zipconvert_5', 'homeowner dummy', 'NUMCHLD', 'INCOME', 'gend
er dummy','WEALTH', 'HV', 'Icmed', 'Icavg', 'IC15', 'NUMPROM', 'RAMN
TALL','MAXRAMNT', 'LASTGIFT', 'totalmonths', 'TIMELAG', 'AVGGIFT', '
TARGET_B','TARGET_D'], dtype='object')
```

- **Set first-column or index for the dataset**
  Using the *set_index* function re-map the 'Row Id' as the first column for the dataset.

  Output



	Row Id.	zipconvert_2	zipconvert_3	zipconvert_4	zipconvert_5	homeowner dummy	NUMCHLD	INCOME	gender dummy	WEALTH	...	IC15	NUMPROM
Row Id													
1	17	0	1	0	0	1	1	5	1	9	...	1	74
2	25	1	0	0	0	1	1	1	0	7	...	4	46
3	29	0	0	0	1	0	2	5	1	8	...	13	32
4	38	0	0	0	1	1	1	3	0	4	...	4	94
5	40	0	1	0	0	1	1	4	0	8	...	7	20

Data Analysis

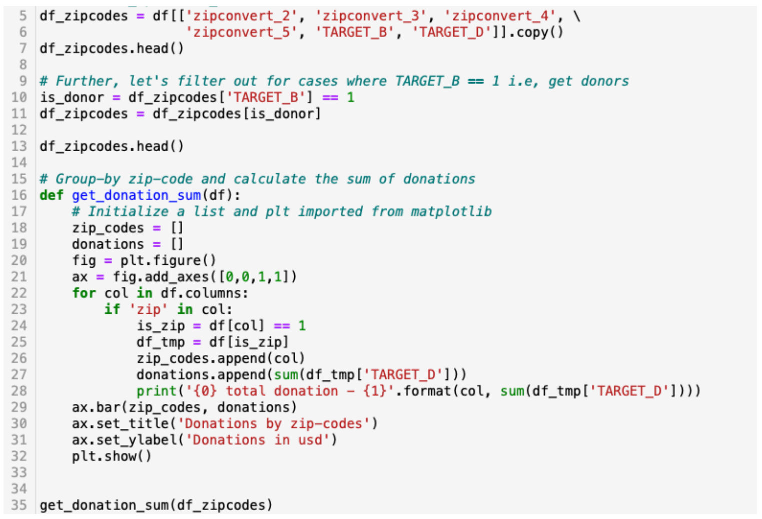
The analysis is mostly directed to find trends and strong correlations to certain people contributing more due to availability of certain social benefits and conditions.

NOTE
At each step, a portion of the dataset that seemed relevant was trimmed and placed in a new data-frame to make analysis simple and crisp.

1. **Donation sum grouped by certain zip-codes/locations**
   Unit of analysis: Donation sum grouped by zip-code

   As a first step, total donations grouped by zip-codes seemed an obvious start. This would set the stage to determine if certain factors within those locations were more favorable for people to make donations often or of higher denominations. Below is the code to achieve this.

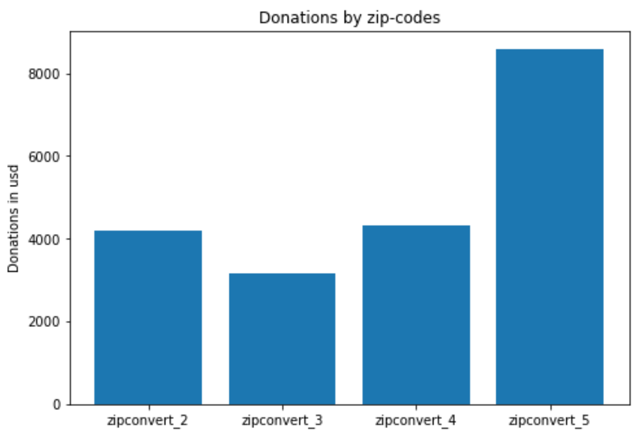
   In line 5-6, a new data-frame *df_zipcodes* was copied from the master data-frame *df* comprising the zipconvert columns along-with the TARGET-B/TARGET-D columns. Also, a simple function *get_donation_sum* to walk the columns, calculate and plot (using matplotlib) the sum of donations was initialized and called.



```python
df_zipcodes = df[['zipconvert_2', 'zipconvert_3', 'zipconvert_4', \
                  'zipconvert_5', 'TARGET_B', 'TARGET_D']].copy()
df_zipcodes.head()

# Further, let's filter out for cases where TARGET_B == 1 i.e, get donors
is_donor = df_zipcodes['TARGET_B'] == 1
df_zipcodes = df_zipcodes[is_donor]

df_zipcodes.head()

# Group-by zip-code and calculate the sum of donations
def get_donation_sum(df):
    # Initialize a list and plt imported from matplotlib
    zip_codes = []
    donations = []
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    for col in df.columns:
        if 'zip' in col:
            is_zip = df[col] == 1
            df_tmp = df[is_zip]
            zip_codes.append(col)
            donations.append(sum(df_tmp['TARGET_D']))
            print('{0} total donation - {1}'.format(col, sum(df_tmp['TARGET_D'])))
    ax.bar(zip_codes, donations)
    ax.set_title('Donations by zip-codes')
    ax.set_ylabel('Donations in usd')
    plt.show()


get_donation_sum(df_zipcodes)
```

Output

```
zipconvert_2 total donation - 4194.37
zipconvert_3 total donation - 3172.0
zipconvert_4 total donation - 4314.5
zipconvert_5 total donation - 8597.92
```



Conclusion
zipconvert_5 had the maximum donations in USD, 8597.92 significantly higher than the other locations.

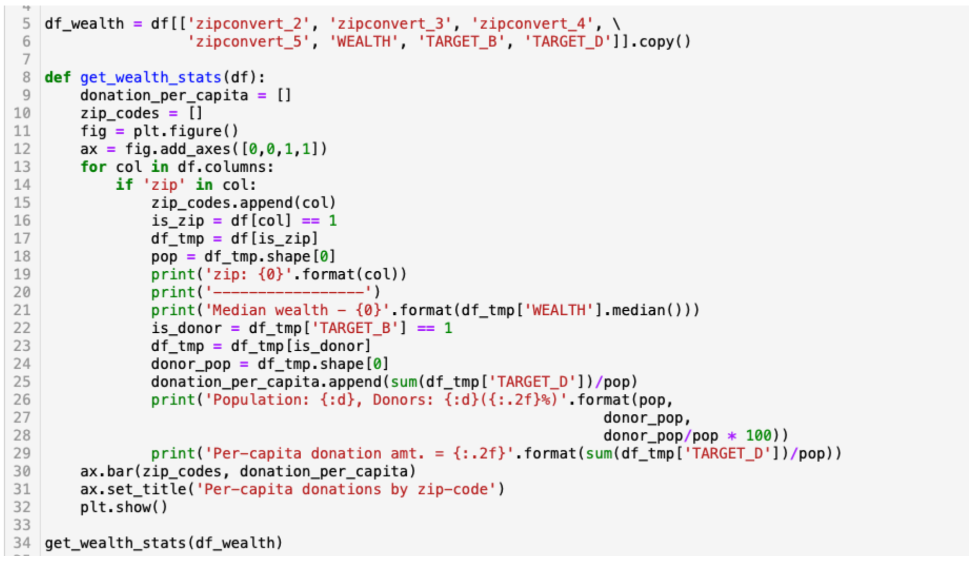
2. Explore wealth-index, and other factors

Unit of analysis: Wealth-index

Based on the results from the previous section wealth-index min, max, mean and median values were derived from the dataset. A new data-frame *df_wealth* additionally comprising the *WEALTH* column was copied from the master data-frame *df.*

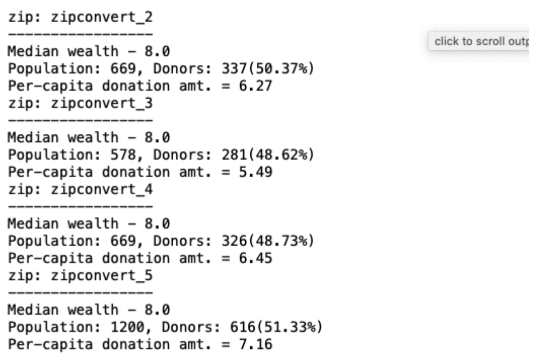
Further, it was important to determine if zipconvert_5 was more densely populated, which lead to higher donation sum. A new metric of *Per-capita donations per zip-code* was introduced to strengthen findings.

The code is presented below. The stats are encompassed in a function *get_wealth_stats* which additionally also plots visualizations

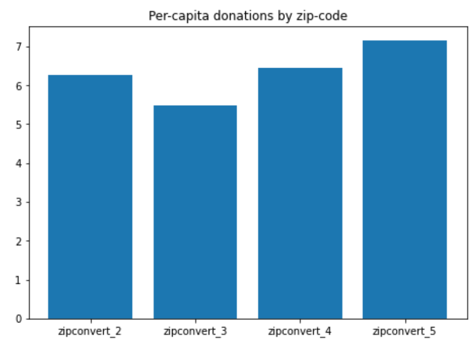


```python
df_wealth = df[['zipconvert_2', 'zipconvert_3', 'zipconvert_4', \
                'zipconvert_5', 'WEALTH', 'TARGET_B', 'TARGET_D']].copy()

def get_wealth_stats(df):
    donation_per_capita = []
    zip_codes = []
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    for col in df.columns:
        if 'zip' in col:
            zip_codes.append(col)
            is_zip = df[col] == 1
            df_tmp = df[is_zip]
            pop = df_tmp.shape[0]
            print('zip: {0}'.format(col))
            print('-----------------')
            print('Median wealth - {0}'.format(df_tmp['WEALTH'].median()))
            is_donor = df_tmp['TARGET_B'] == 1
            df_tmp = df_tmp[is_donor]
            donor_pop = df_tmp.shape[0]
            donation_per_capita.append(sum(df_tmp['TARGET_D'])/pop)
            print('Population: {:d}, Donors: {:d}({:.2f}%)'.format(pop,
                                                                    donor_pop,
                                                                    donor_pop/pop * 100))
            print('Per-capita donation amt. = {:.2f}'.format(sum(df_tmp['TARGET_D'])/pop))
    ax.bar(zip_codes, donation_per_capita)
    ax.set_title('Per-capita donations by zip-code')
    plt.show()

get_wealth_stats(df_wealth)
```

Output



```
zip: zipconvert_2
-----------------
Median wealth - 8.0
Population: 669, Donors: 337(50.37%)
Per-capita donation amt. = 6.27
zip: zipconvert_3
-----------------
Median wealth - 8.0
Population: 578, Donors: 281(48.62%)
Per-capita donation amt. = 5.49
zip: zipconvert_4
-----------------
Median wealth - 8.0
Population: 669, Donors: 326(48.73%)
Per-capita donation amt. = 6.45
zip: zipconvert_5
-----------------
Median wealth - 8.0
Population: 1200, Donors: 616(51.33%)
Per-capita donation amt. = 7.16
```



Conclusion
This proves that the donation sum in zipconvert_5 is higher likely due to a higher population (zipconvert_5 has 1200 people which is significantly more than the others). The percentage of people making donations however is comparable and is ~50%.

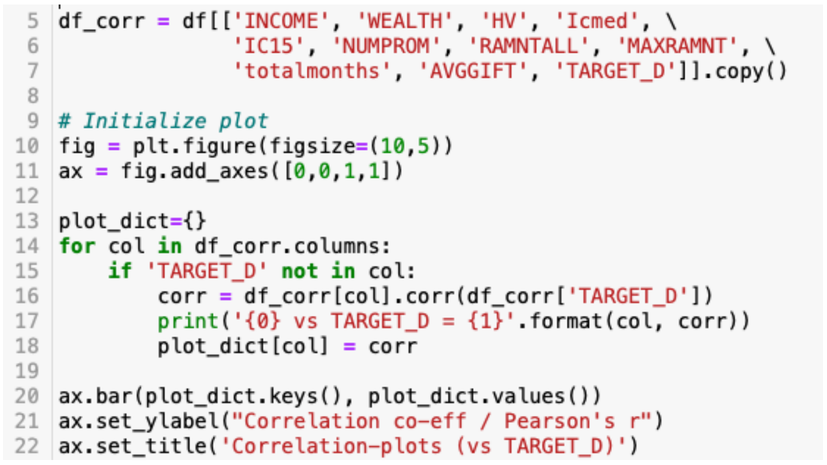
Following are the other key take-aways:
- min (0), max (9), median (8) wealth-index across 4 locations were comparable
- donation per-capita (zip-code) in '5' is 11%, 30% and 14% higher than '4', '3' & '2' respectively

3. Find key correlations

Unit of analysis: Determine based on correlation analysis

Conventionally, we could use the Parsimonious model and determine the most correlated parameters when using the linear regression model. However, we will run this manually on a few parameters using the *corr()* function in pandas.

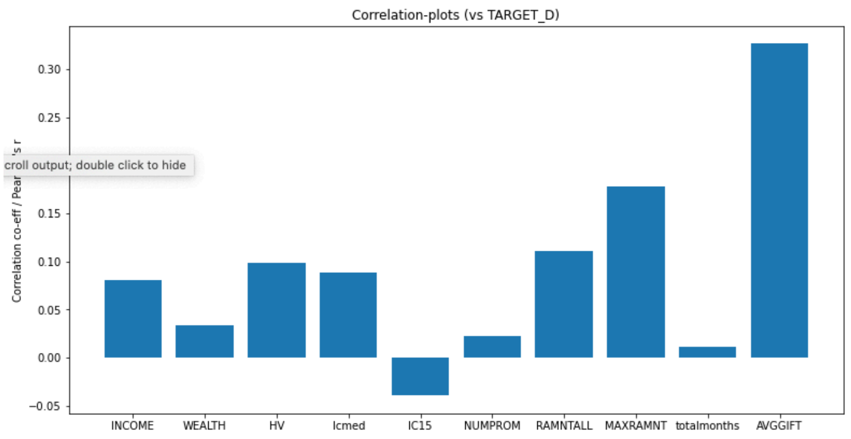
The code below shows how this was achieved



```python
df_corr = df[['INCOME', 'WEALTH', 'HV', 'Icmed', \
              'IC15', 'NUMPROM', 'RAMNTALL', 'MAXRAMNT', \
              'totalmonths', 'AVGGIFT', 'TARGET_D']].copy()

# Initialize plot
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])

plot_dict={}
for col in df_corr.columns:
    if 'TARGET_D' not in col:
        corr = df_corr[col].corr(df_corr['TARGET_D'])
        print('{0} vs TARGET_D = {1}'.format(col, corr))
        plot_dict[col] = corr

ax.bar(plot_dict.keys(), plot_dict.values())
ax.set_ylabel("Correlation co-eff / Pearson's r")
ax.set_title('Correlation-plots (vs TARGET_D)')
```

Output

```
INCOME vs TARGET_D = 0.08032548338264088
WEALTH vs TARGET_D = 0.03364409638488519
HV vs TARGET_D = 0.09893176886948857
Icmed vs TARGET_D = 0.08868927276618381
IC15 vs TARGET_D = -0.03959393387635995
NUMPROM vs TARGET_D = 0.02227075151408228
RAMNTALL vs TARGET_D = 0.11061262841820582
MAXRAMNT vs TARGET_D = 0.17810380290309843
totalmonths vs TARGET_D = 0.01115897079293302
AVGGIFT vs TARGET_D = 0.3266346106284295

Text(0.5, 1.0, 'Correlation-plots (vs TARGET_D)')
```



Conclusion
AVGGIFT among all the variables has the highest correlation to TARGET_D at 0.3. Ideally, we would like the correlation to be closer to 1.0 to show strong correlation.

NOTE
Strong correlation need not mean strong causation but, is still a start. Likely, there is more complex relationship between variables and TARGET_D and this needs to be explored with other machine-learning techniques.

Final Conclusion

The analysis presented in the sections earlier can be summarized as the following:

- one of the locations (zipconvert_5) had a higher population and per-capita contribution at best 30% higher than zipconvert_3
- percentage of people within locations making donations was comparable
- WEALTH index metrics in each location were comparable
    - no strong correlations between parameters and TARGET_D
    - AVGGIFT showed to have a correlation co-eff of 0.3

The problem is not very clearly defined, but the key take-away here would be to gather more details and investigate around why zipconvert_3 had lower donation sum despite comparable metrics.

Description of code

The code was written in entirety in Jupyter Notebook and the corresponding file(.ipynb) is being attached as part of the deliverable. It comprises 4x sub-sections:

1. Data-loading, exploration and clean-up steps
2. The other 3x sections delve into data-analysis

Libraries or packages: *Pandas* for data-analysis and *matplotlib* for visualization

Pandas data-frames were extremely useful in loading the data from csv and running mathematical operations. At each stage the data was sliced and copied into new data-frames to keep things simple at lower scale. Comments, including introductions and conclusions are added to make the code more readable.