TDS2101 - INTRODUCTION TO DATA SCIENCE

Lecture:
TC1V

Project Title:
Financial Crimes Enforcement Network (FinCEN)

Name
Aw Yew Lim
Leong Jean Cheong
Chan Jun Yang

Table of Contents

1.0 Introduction

FinCEN refers to the Financial Crimes Enforcement Network, it is a bureau of the United States Department of the Treasury that collects and analyzes information about financial transactions in order to combat domestic and international money laundering, terrorist financing, and other financial crimes.

1.1 Problem Statement with Motivation

The problem that we found and intend to solve from this dataset is the contributing factors of suspicious transactions. The motivation behind this problem is the incident of the 1Malaysia Development Berhad (1MDB) scandal that occurred in Malaysia in 2015. 1MDB is a Malaysian state fund which was established in 2009 to promote development through foreign investment and partnerships. The fund was one of the biggest corruption scandals in the world where the US justice department believes that the scandal is more than 4.5 billion US dollars. Into the details, vast sums were borrowed via government bonds and then siphoned into bank accounts in Switzerland, Singapore, and United States. Therefore, we would like to know what factors will make a transaction suspicious and explore their relationships as well as building model to tackle this problem.

1.2 Questions

There are some questions that we are facing and would like to answer by undergoing the data science process. The type for each question is descriptive, exploratory and predictive. The questions are listed below:

1. Descriptive:
    - What is the total number and amount of suspicious transactions in each country?
2. Exploratory:
    - What kind of relationships are there between the attributes?
3. Predictive:
    - Can we predict if a transaction is suspicious based on its attributes?

For the descriptive question, we are able to know the total number of the suspicious transactions of each country in each year. This is because we can select countries with highest number of suspicious transactions and compare them to each other to see whether there are some common factors that make their ranking so high. We also get to do some analysis on those countries with higher or lower number of suspicious transactions to figure out the contributing factors and this is related to our exploratory questions. In exploratory question, we are going to find out the relationships between all the attributes in the main and supplementary dataset by doing data analysis. We also tend to find out the pattern between all attributes by

identifying the association between the attributes by using data mining technique. Last but not least, we can build a model that predicts whether a transaction in future is suspicious or not by using machine learning algorithms based on the contributing factors that we discovered from previous questions. Due to our limited dataset, we can have one-class classification approach.

1.3 Dataset

There are two datasets given in this project which are the transaction map and the bank connections. Other than that, we have two additional datasets called ‘countries of the world.csv’. One of the datasets is taken from Kaggle and will give more information about the countries that are involved in suspicious transactions. We will use features from this dataset like population of each country to do further analysis on the factor of making a transaction suspicious. Most of the data work or data modelling will be done with transaction map dataset and other dataset will be stand by as supplementary information. Table below will give some explanation about all the datasets.

Dataset Name	Resource
transaction map	Record of transaction from originator country to beneficiary country
bank connections	Connection between filer and entity bank
countries of the world	Information on population, region, area size, infant mortality and more
Income Group	Information on income group of each country

1.3.1 Transaction Map

Transaction map dataset recorded attributes that manage to assist us in our project outcome. Below visual are the attributes of the dataset:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4507 entries, 0 to 4506
Data columns (total 16 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   id                        4507 non-null   int64  
 1   icij_sar_id               4507 non-null   int64  
 2   filer_org_name_id         4507 non-null   object 
 3   filer_org_name            4507 non-null   object 
 4   begin_date                4501 non-null   object 
 5   end_date                  4501 non-null   object 
 6   originator_bank_id        4507 non-null   object 
 7   originator_bank           4507 non-null   object 
 8   originator_bank_country   4507 non-null   object 
 9   originator_iso            4507 non-null   object 
 10  beneficiary_bank_id       4507 non-null   object 
 11  beneficiary_bank          4507 non-null   object 
 12  beneficiary_bank_country  4507 non-null   object 
 13  beneficiary_iso           4507 non-null   object 
 14  number_transactions       4396 non-null   float64
 15  amount_transactions       4507 non-null   float64
dtypes: float64(2), int64(2), object(12)
```

All the attributes may be the key of the element to perform different results. Filer_org is the original bank who filed the transaction. Of course, the beginning and the end of the transaction must be recorded as a proof when the transaction is made. Besides, in order for a transaction to exist there must be an originator who transfers the money to the beneficiary bank. Certainly, nevertheless the number and amount of the transactions who will record the significant record to find out the suspicious transaction.

1.3.2 Bank Connection

Bank connections are a dataset that records connections between 2 banks which are filer and entity bank. Below visual are the attributes of the dataset:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5498 entries, 0 to 5497
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   icij_sar_id        5498 non-null   int64 
 1   filer_org_name_id  5498 non-null   object
 2   filer_org_name     5498 non-null   object
 3   entity_b_id        5498 non-null   object
 4   entity_b           5498 non-null   object
 5   entity_b_country   5498 non-null   object
 6   entity_b_iso_code  5498 non-null   object
dtypes: int64(1), object(6)
```

1.3.3 Country of the World

Countries of the world dataset records every country's basic information for example population, GDP, and etc. Below visual are the attributes of the dataset:

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 20 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Country                             227 non-null    object 
 1   Region                              227 non-null    object 
 2   Population                          227 non-null    int64  
 3   Area (sq. mi.)                      227 non-null    int64  
 4   Pop. Density (per sq. mi.)          227 non-null    object 
 5   Coastline (coast/area ratio)        227 non-null    object 
 6   Net migration                       224 non-null    object 
 7   Infant mortality (per 1000 births)  224 non-null    object 
 8   GDP ($ per capita)                  226 non-null    float64
 9   Literacy (%)                        209 non-null    object 
 10  Phones (per 1000)                   223 non-null    object 
 11  Arable (%)                          225 non-null    object 
 12  Crops (%)                           225 non-null    object 
 13  Other (%)                           225 non-null    object 
 14  Climate                             205 non-null    object 
 15  Birthrate                           224 non-null    object 
 16  Deathrate                           223 non-null    object 
 17  Agriculture                         212 non-null    object 
 18  Industry                            211 non-null    object 
 19  Service                             212 non-null    object 
dtypes: float64(1), int64(2), object(17)
```

1.3.4 Income Group

Income Group dataset records every country's information for example abbreviation name, region, income group and etc. Below visual are the attributes of the dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Country Code  263 non-null    object 
 1   Region        217 non-null    object 
 2   IncomeGroup   217 non-null    object 
 3   SpecialNotes  94 non-null     object 
 4   TableName     263 non-null    object 
 5   Unnamed: 5    0 non-null      float64
dtypes: float64(1), object(5)
memory usage: 12.5+ KB
```

1.4 Potential Benefits

There are also some potential benefits or impacts of our problem to be solved on a specific target sector or business, or society/nation. For example, tackling financial crimes such as money laundering, able to recover the credibility of the legitimate financial sector which will attract investors back to make investments. Concurrently, the economy of the country will also regain its momentum and grow. It will be easier to tackle financial crimes if we know the contributing factors of making a transaction suspicious. We can make use of Artificial Intelligence to detect suspicious transactions automatically and make human's life easier.

2.0 Data Science Pipeline

After understanding our problem on FinCEN and the questions that we intend to solve, we are going to put those ideas and plans into action. There are five main steps in Data Science pipeline, which is data collection, data preprocessing, data analysis, data mining or data modelling, and data visualization. We are going to solve our problem by walking through the Data Science pipeline. To do so, we will use Jupyter Notebook to work on all the processes that is going to happen.

2.1 Data Collection

With helps of lecturer, we are given two datasets that is shown and explained in previous section to extract findings from them. Luckily, we have also found two supplementary dataset which contains some information of each country from online sources to support our main dataset. However, all the transactions in our dataset are suspicious, which is positive data. Therefore, we have to spend some time on collecting negative data. Unfortunately, we cannot find any negative data from online sources. Therefore, we use our own recent bank transaction records to be the negative data although only 8 records are collected. At this moment, we decide to find out some ways to train models with only positive data, and the 8 negative data will only be used to evaluate our model. The negative data is shown below:

	IncomeGroup	GDP ($ per capita)	duration(days)	number_transactions	amount_transactions	suspicious
0	3	9000	1	1	878	-1
1	4	23700	275	5	8000	-1
2	4	9000	1	1	500	-1
3	3	9000	1	1	100	-1
4	3	9000	1	1	1500	-1
5	4	23700	0	1	1300	-1
6	4	9000	1	1	5000	-1
7	3	9000	1	1	2000	-1

2.2 Data Preprocessing

Data preprocessing is a data science technique that includes transforming raw data into a format that is understandable. Normally, the real-world data is inconsistent, incomplete, or contains a lot of errors and noises, and data preprocessing is a method to solve these problems. Similarly, our main dataset is not clean enough to be used to do further work. Therefore, we are going to do some data cleaning on our main dataset.

First of all, we check whether there is NULL value in our dataset. The result shows that there are 6 null values in both begin_date and end_date columns while there are 111 null values in number_transactions column.

```
df.isnull().sum()
```

```
id                          0
icij_sar_id                 0
filer_org_name_id           0
filer_org_name              0
begin_date                  6
end_date                    6
originator_bank_id          0
originator_bank             0
originator_bank_country     0
originator_iso              0
beneficiary_bank_id         0
beneficiary_bank            0
beneficiary_bank_country    0
beneficiary_iso             0
number_transactions       111
amount_transactions         0
dtype: int64
```

We decide to drop the row with NULL value in begin_date and end_date because it is not possible to find the mean of a date or predict the date. Another reason why we drop them is because we still have enough data to proceed after dropping 6 rows of data with NULL values. However, we cannot drop all rows with null values in number_transactions column because there are 111 NULL values in it. Therefore, we decide to fill the null values with the mean of all values in number_transactions column. After that, we check back the number of null values and the result indicates that there is no NULL value in each column anymore.

```
df = df.dropna(subset = ["begin_date", "end_date"])
```

```
df = df.fillna(df.mean().round())
```

```
df.isnull().sum()
```

```
id                          0
icij_sar_id                 0
filer_org_name_id           0
filer_org_name              0
begin_date                  0
end_date                    0
originator_bank_id          0
originator_bank             0
originator_bank_country     0
originator_iso              0
beneficiary_bank_id         0
beneficiary_bank            0
beneficiary_bank_country    0
beneficiary_iso             0
number_transactions         0
amount_transactions         0
dtype: int64
```

Since our questions are concern on the contributing factors, we will only use some columns instead of using all. Therefore, we decide to drop all the columns that we are not going to use in our main dataset.

```
df.drop(['id', 'icij_sar_id', 'filer_org_name_id', 'filer_org_name', 'originator_bank_id', 'originator_bank',
        'beneficiary_bank_id', 'beneficiary_bank', 'beneficiary_iso'], axis=1, inplace=True)
```

The last problem that we might encounter is duplicate data. Duplicate data can be any record that inadvertently shares data with another record in dataset. The most visible form of duplicate data is a complete carbon copy of another record. It will not be balanced if there is duplicate data in dataset and might cause model training with bias in future. Therefore, we also check if there is any duplicate data in our main dataset. The result shows that there are 67 duplicates data in main dataset, and since it is a very small portions of our main dataset, we decide to drop it off the dataset.

```
df[df.duplicated()].shape[0]
```

```
67
```

```
df = df.drop_duplicates()
```

```
df[df.duplicated()].shape[0]
```

```
0
```

Our main dataset is now cleaned and ready to be used. Next, we are not going to clean our supplementary dataset. We first dropping all the columns that are not going to be used in this project.

```
sup_df.drop(['Pop. Density (per sq. mi.)', 'Coastline (coast/area ratio)', 'Net migration',
             'Infant mortality (per 1000 births)', 'Literacy (%)', 'Phones (per 1000)',
             'Arable (%)', 'Crops (%)', 'Other (%)', 'Climate', 'Birthrate',
             'Deathrate', 'Agriculture', 'Industry', 'Service'], axis=1, inplace=True)
```

Finally, we decide to merge two datasets into one to make ourselves more convenient to use the dataset to do further works. After that, we will only deal with the null values on the merged dataset if there is any.

```
merged_df = df.join(sup_df.set_index('Country'), on='originator_bank_country')
merged_df
```

	begin_date	end_date	originator_bank_country	originator_iso	beneficiary_bank_country	number_transactions	amount_transactions	Region	Populatio
0	25-Mar-15	25-Sep-15	Singapore	SGP	United Kingdom	68.0	56900000.0	ASIA (EX. NEAR EAST)	4492150.
1	30-Mar-15	25-Sep-15	Singapore	SGP	United Kingdom	118.0	116000000.0	ASIA (EX. NEAR EAST)	4492150.
2	5-Jul-12	5-Jul-12	United Kingdom	GBR	Sweden	4.0	5000.0	WESTERN EUROPE	60609153.
3	20-Jun-12	20-Jun-12	United Kingdom	GBR	Sweden	4.0	9990.0	WESTERN EUROPE	60609153.
4	31-May-12	31-May-12	United Kingdom	GBR	Sweden	4.0	12000.0	WESTERN EUROPE	60609153.
...	...	...	...	...	...	...	...	...	...
4502	22-Aug-16	22-Aug-16	Netherlands	NLD	Latvia	1.0	20400000.0	WESTERN EUROPE	16491461.
4503	16-Aug-16	16-Aug-16	Latvia	LVA	Netherlands	1.0	23000000.0	BALTICS	2274735.
4504	16-Aug-16	16-Aug-16	Netherlands	NLD	Latvia	1.0	23416000.0	WESTERN EUROPE	16491461.
4505	14-Jun-16	14-Jun-16	Latvia	LVA	Netherlands	1.0	33300000.0	BALTICS	2274735.
4506	14-Jun-16	14-Jun-16	Netherlands	NLD	Latvia	1.0	33259000.0	WESTERN EUROPE	16491461.

```
merged_df.isnull().sum()
```

```
begin_date                0
end_date                  0
originator_bank_country   0
originator_iso            0
beneficiary_bank_country  0
number_transactions       0
amount_transactions       0
Region                  124
Population              124
Area (sq. mi.)          124
GDP ($ per capita)      124
dtype: int64
```

```
merged_df = merged_df.dropna(subset=['Region'])
```

```
merged_df.isnull().sum()
```

```
begin_date                0
end_date                  0
originator_bank_country   0
originator_iso            0
beneficiary_bank_country  0
number_transactions       0
amount_transactions       0
Region                    0
Population                0
Area (sq. mi.)            0
GDP ($ per capita)        0
dtype: int64
```

Then, we would like to calculate the duration of transaction based on the begin date and end date. We first split the date to year, month and day and use pandas function to calculate the duration.

```
merged_df[["year", "month", "day"]] = merged_df["begin_date"].str.split("-", expand=True)
merged_df[["year2", "month2", "day2"]] = merged_df["end_date"].str.split("-", expand=True)

merged_df[["start"]] = pd.to_datetime( merged_df["year"] + "/" + merged_df["month"] + "/" + merged_df["day"])
merged_df[["end"]] = pd.to_datetime( merged_df["year2"] + "/" + merged_df["month2"] + "/" + merged_df["day2"])

merged_df[["duration(days)"]] = 1 + ((merged_df['end'] - merged_df['start']).dt.days)
merged_df = merged_df.drop(['begin_date', 'end_date', 'year', 'month', 'day','year2','month2','day2','start','end'],axis=1)
merged_df['suspicious'] = 1
merged_df
```

	originator_bank_country	originator_iso	beneficiary_bank_country	IncomeGroup	number_transactions	amount_transactions	Region	Population	Area (sq. mi.)	GDP ($ per capita)	duration(days)	suspicious
0	Singapore	SGP	United Kingdom	4.0	68.0	56900000.0	ASIA (EX. NEAR EAST)	4492150.0	693.0	23700.0	185	1
1	Singapore	SGP	United Kingdom	4.0	118.0	116000000.0	ASIA (EX. NEAR EAST)	4492150.0	693.0	23700.0	180	1
2	United Kingdom	GBR	Sweden	4.0	4.0	5000.0	WESTERN EUROPE	60609153.0	244820.0	27700.0	1	1
3	United Kingdom	GBR	Sweden	4.0	4.0	9990.0	WESTERN EUROPE	60609153.0	244820.0	27700.0	1	1
4	United Kingdom	GBR	Sweden	4.0	4.0	12000.0	WESTERN EUROPE	60609153.0	244820.0	27700.0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
4502	Netherlands	NLD	Latvia	4.0	1.0	20400000.0	WESTERN EUROPE	16491461.0	41526.0	28600.0	1	1
4503	Latvia	LVA	Netherlands	4.0	1.0	23000000.0	BALTICS	2274735.0	64589.0	10200.0	1	1
4504	Netherlands	NLD	Latvia	4.0	1.0	23416000.0	WESTERN EUROPE	16491461.0	41526.0	28600.0	1	1
4505	Latvia	LVA	Netherlands	4.0	1.0	33300000.0	BALTICS	2274735.0	64589.0	10200.0	1	1
4506	Netherlands	NLD	Latvia	4.0	1.0	33259000.0	WESTERN EUROPE	16491461.0	41526.0	28600.0	1	1

Now, we would like to know the income group of each country in this step of data. Therefore, we read the income group dataset and merge them into one. After that, we rank the income group into category [1,2,3,4].

```
income_group_df = pd.read_csv('Income_Group.csv')
income_group_df
```

	Country Code	Region	IncomeGroup	SpecialNotes	TableName	Unnamed: 5
0	ABW	Latin America & Caribbean	High income	NaN	Aruba	nan
1	AFG	South Asia	Low income	NaN	Afghanistan	nan
2	AGO	Sub-Saharan Africa	Lower middle income	NaN	Angola	nan
3	ALB	Europe & Central Asia	Upper middle income	NaN	Albania	nan
4	AND	Europe & Central Asia	High income	NaN	Andorra	nan
...	...	...	...	...	...	...
258	XKX	Europe & Central Asia	Upper middle income	NaN	Kosovo	nan
259	YEM	Middle East & North Africa	Low income	NaN	Yemen, Rep.	nan
260	ZAF	Sub-Saharan Africa	Upper middle income	Fiscal year end: March 31; reporting period fo...	South Africa	nan
261	ZMB	Sub-Saharan Africa	Lower middle income	National accounts data were rebased to reflect...	Zambia	nan
262	ZWE	Sub-Saharan Africa	Lower middle income	NaN	Zimbabwe	nan

```
income_group_df = income_group_df[['Country Code', 'IncomeGroup']]
merged_df = merged_df.join(income_group_df.set_index('Country Code'), on='originator_iso')
```

```
merged_df['IncomeGroup'] = merged_df['IncomeGroup'].map({'Low income': 1, 'Lower middle income': 2, 'Upper middle in
```

We discover that there are some NULL values in IncomeGroup columns, therefore we decide to drop them out.


```
income_group_df = income_group_df[['Country Code', 'IncomeGroup']]
merged_df = merged_df.join(income_group_df.set_index('Country Code'), on='originator_iso')
```

```
merged_df['IncomeGroup'] = merged_df['IncomeGroup'].map({'Low income': 1, 'Lower middle income': 2, 'Upper middle in
```


2.3 Data Analysis

At first, we are going to find answer for several descriptive questions in order to get better understanding on our data.

Question 1: What are the number of transactions, amount of transactions, and the average amount of transactions for each countries?

```
total = merged_df[['originator_bank_country', 'number_transactions', 'amount_transactions']]
total = total.groupby(['originator_bank_country'], as_index=False).sum()
total['average_amount_transactions'] = total['amount_transactions'] / total['number_transactions']
total
```

	originator_bank_country	number_transactions	amount_transactions	average_amount_transactions
0	Afghanistan	65.0	4.720310e+07	7.262015e+05
1	Andorra	7.0	2.867110e+06	4.095871e+05
2	Angola	11.0	1.645580e+06	1.495981e+05
3	Armenia	2.0	3.750022e+05	1.875011e+05
4	Australia	160.0	1.680031e+08	1.050019e+06
...	...	...	...	...
101	United States	1199.0	1.188888e+09	9.915666e+05
102	Uruguay	26.0	6.538825e+06	2.514933e+05
103	Uzbekistan	7.0	5.090787e+06	7.272553e+05
104	Vietnam	3.0	7.221572e+06	2.407191e+06
105	Zambia	103.0	1.140362e+07	1.107147e+05

106 rows × 4 columns

Question 2: Which countries contributes to the majority of suspicious transactions?

```
majority_countries = total[['originator_bank_country', 'amount_transactions']]
majority_countries = majority_countries.sort_values(by=['amount_transactions'], ignore_index=True)
majority_countries = majority_countries
majority_countries
```

	originator_bank_country	amount_transactions
0	Kenya	3.750000e+03
1	Malta	4.780000e+03
2	Costa Rica	6.200000e+03
3	Kuwait	1.160227e+04
4	Sri Lanka	2.738320e+04
...	...	...
101	Russia	3.413378e+09
102	United Kingdom	3.558382e+09
103	Switzerland	4.105413e+09
104	Netherlands	4.968629e+09
105	Latvia	6.350550e+09

106 rows × 2 columns

Question 3: What are the number of transactions, amount of transactions, and the average amount of transactions for each region?

```
regional_total = merged_df[['Region', 'number_transactions', 'amount_transactions']]
regional_total = regional_total.groupby('Region', as_index=False).sum()
regional_total['average_amount_transactions'] = regional_total['amount_transactions'] / regional_total['number_transactions']
regional_total
```

	Region	number_transactions	amount_transactions	average_amount_transactions
0	ASIA (EX. NEAR EAST)	3179.0	4.788211e+09	1.506200e+06
1	BALTICS	2626.0	6.492262e+09	2.472301e+06
2	C.W. OF IND. STATES	3435.0	3.842508e+09	1.118634e+06
3	EASTERN EUROPE	561.0	4.289491e+08	7.646151e+05
4	LATIN AMER. & CARIB	456.0	1.108693e+09	2.431344e+06
5	NEAR EAST	1834.0	8.879530e+08	4.841619e+05
6	NORTHERN AFRICA	18.0	1.094608e+07	6.081157e+05
7	NORTHERN AMERICA	1298.0	1.206487e+09	9.294967e+05
8	OCEANIA	164.0	1.680814e+08	1.024887e+06
9	SUB-SAHARAN AFRICA	450.0	4.370356e+08	9.711902e+05
10	WESTERN EUROPE	4174.0	1.569988e+10	3.761351e+06

Question 4: Which regions contributes to majority of suspicious transactions?

```
majority_region = regional_total[['Region', 'amount_transactions']]
majority_region = majority_region.sort_values(by='amount_transactions', ascending=False, ignore_index=True)
majority_region
```

	Region	amount_transactions
0	WESTERN EUROPE	1.569988e+10
1	BALTICS	6.492262e+09
2	ASIA (EX. NEAR EAST)	4.788211e+09
3	C.W. OF IND. STATES	3.842508e+09
4	NORTHERN AMERICA	1.206487e+09
5	LATIN AMER. & CARIB	1.108693e+09
6	NEAR EAST	8.879530e+08
7	SUB-SAHARAN AFRICA	4.370356e+08
8	EASTERN EUROPE	4.289491e+08
9	OCEANIA	1.680814e+08
10	NORTHERN AFRICA	1.094608e+07

Now, we going to answer our exploratory question which is to find out what kind of relationships are there between the attributes by finding out the correlation between the attributes.

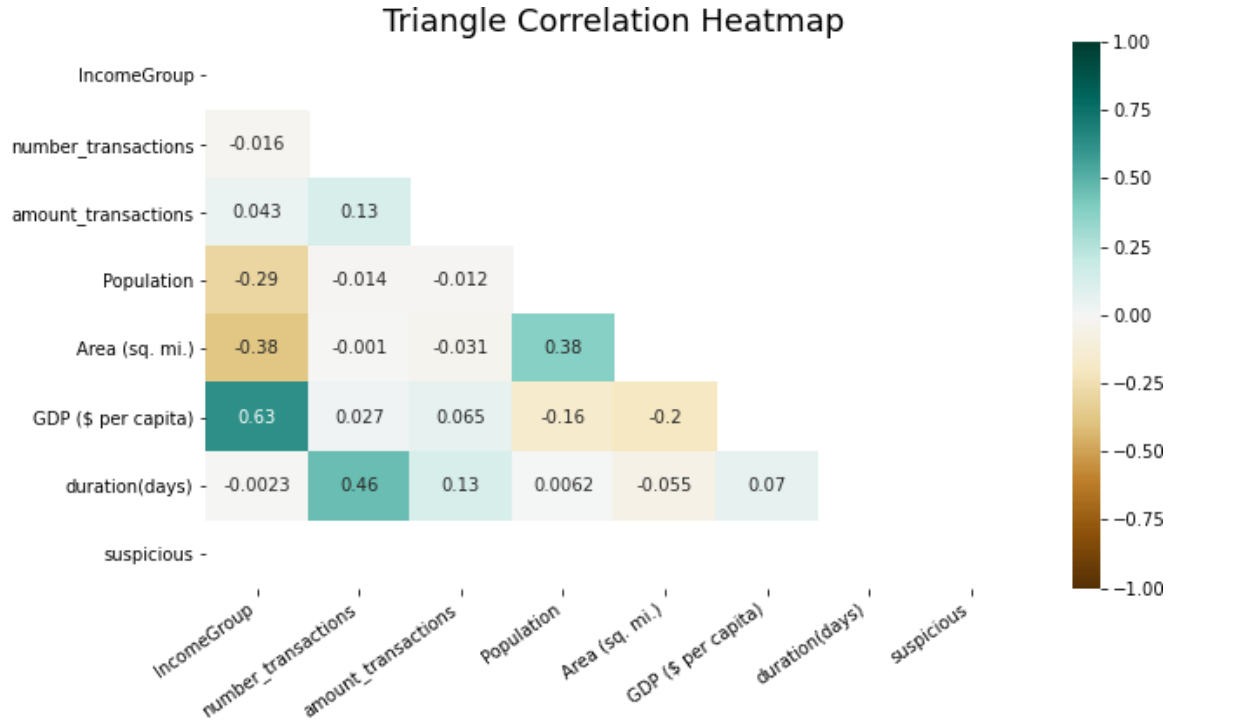
```
cor = merged_df.corr()
mask = np.triu(np.ones_like(cor, dtype=np.bool))

plt.figure(figsize=(10,6))

heatmap = sns.heatmap(cor, mask=mask, vmin=-1, vmax=1, annot=True, cmap='BrBG')

heatmap.set_title('Triangle Correlation Heatmap', fontsize=18)
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=20, ha='right')

plt.tight_layout()
plt.savefig('heatmap.png')
```



From the chart above, we observe that the duration has a minimal relationship with number of transactions. Other than that, income group has a moderate relationship with GDP and minimal relationship with Population and the size of country (Area). As for remaining attributes, they are having no relationship with each other.

2.4 Data Modelling

For our predictive question, we are going to answer whether we can predict if a transaction is suspicious or not based on its attributes. If the answer is yes, it can be used to tackle financial crimes such as money laundering, and is able to recover the credibility of the legitimate financial sector which will attract investors back to make investments. To answer this question, we will train a machine learning model with the data

that we have and evaluate its performance. However, we have a lot of positive data, which is suspicious transaction, and a very minor part of negative data to train the model. It is really imbalanced dataset and therefore we decide to use One-Class SVM to perform one class classification to tackle this problem. For example, if we take binary classification, SVM tries to find best possible space between A and B. If there is only one class A, model tries to create a boundary around it and classify, and this is exactly our case. We have only the positive class of data and that will be the reason why we approach to on class classification.

Firstly, we import all necessary library to Jupyter Notebook and read the negative data into a data frame.

```
from sklearn.model_selection import train_test_split
from sklearn.svm import OneClassSVM
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
self_df = pd.read_csv('self_bank_transaction.csv')
self_df
```

We then filter out the columns of positive data and negative data that we are not going to use and append them together followed by splitting them to X and y. X is the features and y is the label. At the same time, we set the 'IncomeGroup' and 'suspicious' column to category type because they are categorical data.

```
X = merged_df[['IncomeGroup','GDP ($ per capita)', 'duration(days)', 'number_transactions', 'amount_transactions']]
y = merged_df['suspicious']
X = X.append(X_self[['IncomeGroup','GDP ($ per capita)', 'duration(days)',
                     'number_transactions', 'amount_transactions']], ignore_index=True)
y = y.append(X_self['suspicious'], ignore_index=True).astype('category')
X['IncomeGroup'] = X['IncomeGroup'].astype('category')
```

After that, we split the data into training set and test set and fits the model on the data from the positive class only in the training set. The result is shown below:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=2)
model = OneClassSVM(gamma='scale', nu=0.001)
model.fit(X_train[y_train == 1])
print("Training Accuracy: ", accuracy_score(y_train, model.predict(X_train)))
print("Test Accuracy: ", accuracy_score(y_test, model.predict(X_test)))
print("Precision Score: ", precision_score(y_train, model.predict(X_train)))
print("Recall Score: ", recall_score(y_train, model.predict(X_train)))
print("F1-score: ", f1_score(y_train, model.predict(X_train), pos_label=-1))
```

```
Training Accuracy:  0.9899165507649513
Test Accuracy:  0.9901199717713479
Precision Score:  0.9985969835145563
Recall Score:  0.9912952646239555
F1-score:  0.0
```

We are satisfied to see the good training and test accuracy. However, they cannot be used as evaluation metrics in one class classification due to our imbalanced data. In this case, we will use F-measure score, which is the harmonic mean of precision and recall. We can calculate the F-measure using the *f1_score()* function and specify the label of the negative class as -1 via the "*pos_label*" argument. An

F1 score of 0 is achieved. Since our majority class of data is positive, to test the performance of our model, we have also used our own negative data and the result is pretty impressive.

```
X_self = self_df[['IncomeGroup','GDP ($ per capita)','duration(days)', 'number_transactions', 'amount_transactions']]
model.predict(X_self)
```

```
array([1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

3.0 Conclusion

Throughout the analysis, we found that most of the suspicious transactions are from regions of Western Europe, Baltics, Asia, Commonwealth of Independent States, and Northern America. Besides, our machine learning model are able to predict if a transaction is suspicious via its attributes. On the opposite, we are only able to find the relationship between each attribute, and failed to find any factors that contribute to suspicious transaction. In future, if we are able to obtain more useful data, we could improve our machine learning model to be more accurate on predicting especially when the amount of transaction is small.

Besides that, we also faced some challenges when completing this project:

- Limited information – The main dataset has limited information to help us find out insights for our main problem. Meanwhile, it is pretty hard to source for any supplementary dataset that is useful for our cases.
- Time – As we facing the limited information, we spend a lot of time to source for supplementary data set. Also, having assignments from other subject, we have to manage our time very carefully to complete our jobs in time.