Clustering Analysis for Start-up Companies

Objective

- **Clustering VC funds based on their existing investments.**

  The aim is to derive the relationship among the companies and their investors. There was a need to join the given data sets, perform multiple pre-processing activities, exploratory data analysis to develop clustering models using multiple algorithms.

The Data

- Three CSV files, given.
  - Objects.csv : Represents the transactional data about various companies
  - Funds.csv : Represents details about funding
  - Investments.csv : Holds the key information to join the Objects and Funds

Libraries Used

Library	Description
Pandas	for file operations
Numpy	for mathematical operations
Seaborn, Matplotlib	for graphical plotting
Networkx	for network graphs
Sklearn	for data processing

Load and Pre-process Objects file

- Load and explore the data : Objects.csv
    1. Found 4,62,651 rows and 40 columns
    2. After analysing the data in detail, we need to remove 26 least required columns that hold very large number of null values and carrying forward with 14 other columns like “homepage_url”, “twitter_username”, “logo_url”, etc. This is done as per domain knowledge and number of null values observed in a column.
        - We used df_objects.info() to find the no. of null rows
        - And used drop() command to remove the least essential columns

Identifying Non-null count in all columns (40)

```
RangeIndex: 462651 entries, 0 to 462650
Data columns (total 40 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   id                    462651 non-null  object
 1   entity_type           462651 non-null  object
 2   entity_id             462651 non-null  int64
 3   parent_id             27715 non-null   object
 4   name                  462649 non-null  object
 5   normalized_name       462620 non-null  object
 6   permalink             462651 non-null  object
 7   category_code         123186 non-null  object
 8   status                462651 non-null  object
 9   founded_at            100441 non-null  object
 10  closed_at             2809 non-null    object
 11  domain                174942 non-null  object
 12  homepage_url          174942 non-null  object
 13  twitter_username      126089 non-null  object
 14  logo_url              208850 non-null  object
 15  logo_width            462651 non-null  int64
 16  logo_height           462651 non-null  int64
 17  short_description     7617 non-null    object
 18  description           95005 non-null   object
 19  overview              235235 non-null  object
 20  tag_list              106496 non-null  object
 21  country_code          95043 non-null   object
 22  state_code            54760 non-null   object
 23  city                  90684 non-null   object
 24  region                462651 non-null  object
 25  first_investment_at   16956 non-null   object
 26  last_investment_at    16956 non-null   object
 27  investment_rounds     462651 non-null  int64
 28  invested_companies    462651 non-null  int64
 29  first_funding_at      31507 non-null   object
 30  last_funding_at       31507 non-null   object
 31  funding_rounds        462651 non-null  int64
 32  funding_total_usd     462651 non-null  float64
 33  first_milestone_at    100358 non-null  object
 34  last_milestone_at     100358 non-null  object
 35  milestones            462651 non-null  int64
 36  relationships         462651 non-null  int64
 37  created_by            339486 non-null  object
 38  created_at            462591 non-null  object
 39  updated_at            462651 non-null  object
dtypes: float64(1), int64(8), object(31)
memory usage: 141.2+ MB
```

Remaining columns (14)

```
Data columns (total 14 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   id                   462651 non-null  object
 1   entity_type          462651 non-null  object
 2   name                 462649 non-null  object
 3   category_code        123186 non-null  object
 4   status               462651 non-null  object
 5   founded_at           100441 non-null  object
 6   country_code         95043 non-null   object
 7   first_investment_at  16956 non-null   object
 8   last_investment_at   16956 non-null   object
 9   investment_rounds    462651 non-null  int64
 10  invested_companies   462651 non-null  int64
 11  funding_rounds       462651 non-null  int64
 12  funding_total_usd    462651 non-null  float64
 13  milestones           462651 non-null  int64
dtypes: float64(1), int64(4), object(9)
memory usage: 49.4+ MB
```

3. Identified the following important categorical columns
    - Status, Category_code, Entity_type, Country_code
    - A sample code snippet of status is given below

```
df_objects['status'].value_counts()
```

```
operating      443663
acquired         9394
live             4349
closed           2773
ipo              1134
beta              780
development       226
private           219
alpha             113
Name: status, dtype: int64
```

- Loaded **Investments.csv**
  - Found 80,902 rows and 6 columns
  - This file is mainly used to connecting objects and funds. Keeping the below two keys, we have removed the other 4 least required columns.

```
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   funded_object_id    80902 non-null  object
 1   investor_object_id  80902 non-null  object
dtypes: object(2)
memory usage: 1.2+ MB
```

- As per the above meta data, we found a funded_object_id is to be joined with **id** column of objects data frame. Renamed the id column of objects to **funded_object_id.** This is done as a best practice, which also makes us join the tables easily.
- Created a new data frame using Objects and investments data frames. Below is the screenshot of the merged data frame metadata

```
Int64Index: 80570 entries, 0 to 80569
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   funded_object_id     80570 non-null  object
 1   investor_object_id   80570 non-null  object
 2   entity_type          80570 non-null  object
 3   name                 80570 non-null  object
 4   category_code        78704 non-null  object
 5   status               80570 non-null  object
 6   founded_at           68590 non-null  object
 7   country_code         77875 non-null  object
 8   first_investment_at  921 non-null    object
 9   last_investment_at   921 non-null    object
 10  investment_rounds    80570 non-null  int64
 11  invested_companies   80570 non-null  int64
 12  funding_rounds       80570 non-null  int64
 13  funding_total_usd    80570 non-null  float64
 14  milestones           80570 non-null  int64
dtypes: float64(1), int64(4), object(10)
memory usage: 9.8+ MB
```

- Loaded the **funds.csv**
  - Found 1564 rows and 11 columns
  - After analysing the content, decided to drop 7 least relevant columns, and carrying forward with remaining 4

```
df_funds.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1564 entries, 0 to 1563
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    1564 non-null   int64
 1   fund_id               1564 non-null   int64
 2   object_id             1564 non-null   object
 3   name                  1564 non-null   object
 4   funded_at             1449 non-null   object
 5   raised_amount         1564 non-null   float64
 6   raised_currency_code  1564 non-null   object
 7   source_url            1272 non-null   object
 8   source_description    1218 non-null   object
 9   created_at            1564 non-null   object
 10  updated_at            1564 non-null   object
dtypes: float64(1), int64(2), object(8)
memory usage: 134.5+ KB
```

```
df_funds.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1564 entries, 0 to 1563
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   object_id             1564 non-null   object
 1   funded_at             1449 non-null   object
 2   raised_amount         1564 non-null   float64
 3   raised_currency_code  1564 non-null   object
dtypes: float64(1), object(3)
memory usage: 49.0+ KB
```

- Object_id of the above data frame is renamed to investor_object_id. This is done after checking the contents of it, and it is related to the investor_object_id of previous merged data frame.
- A new data frame is created with the name df_objects_investments_funds to analyse further
- A total of 71,811 rows and 16 columns is used
- Rows containing null values are also removed from the dataframe as missing values could not be replaced with mean or median values
- Other data corrections issues were not observed. Only string to float / integer may be required in future to analyse the clustering

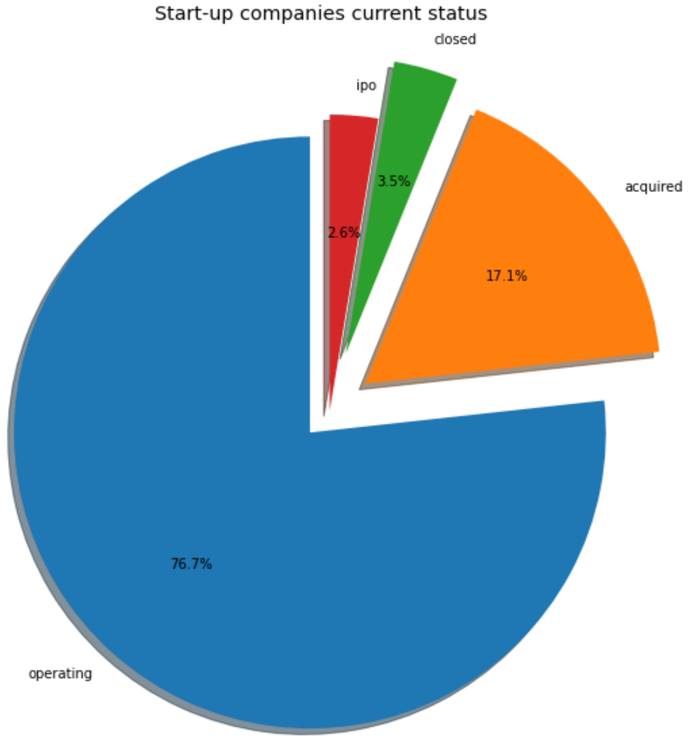
```
df_objects_investments_funds.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 71811 entries, 0 to 71810
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   funded_object_id      71811 non-null  object
 1   investor_object_id    71811 non-null  object
 2   entity_type           71811 non-null  object
 3   name                  71811 non-null  object
 4   category_code         70003 non-null  object
 5   status                71811 non-null  object
 6   founded_at            62685 non-null  object
 7   country_code          70041 non-null  object
 8   investment_rounds     71811 non-null  int64
 9   invested_companies    71811 non-null  int64
 10  funding_rounds        71811 non-null  int64
 11  funding_total_usd     71811 non-null  float64
 12  milestones            71811 non-null  int64
 13  funded_at             69562 non-null  object
 14  raised_amount         71811 non-null  float64
 15  raised_currency_code  71811 non-null  object
dtypes: float64(2), int64(4), object(10)
```

```
df_objects_investments_funds.isna().sum()

funded_object_id        0
investor_object_id      0
entity_type             0
name                    0
category_code           0
status                  0
founded_at              0
country_code            0
investment_rounds       0
invested_companies      0
funding_rounds          0
funding_total_usd       0
milestones              0
funded_at               0
raised_amount           0
raised_currency_code    0
dtype: int64
```

- After removing Null values, data count decreased from 71,811 to 59,446.
- Identified some duplicate entries and removed. After removing duplicates entries also, row count reduced from 59,446 to 37,835

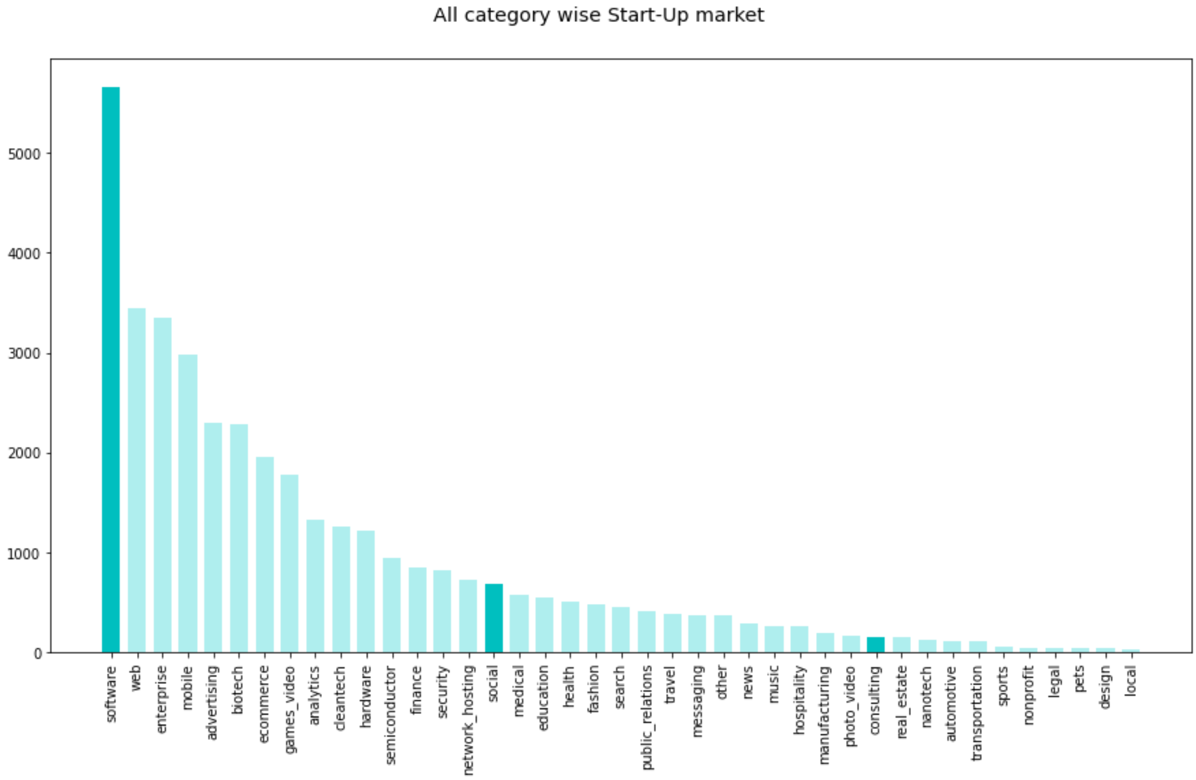
Graphical Exploratory Data Analysis

- Considered the "current status" of the start-up companies with graphical representations (Pie Chart)

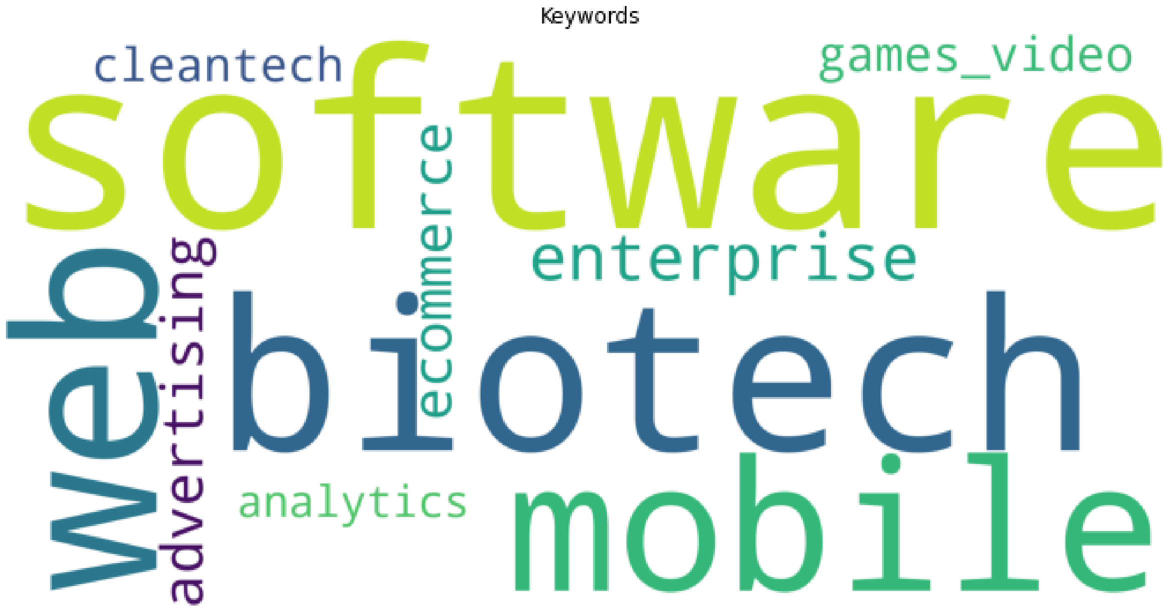


The above graph highlights that 76.7% percentage of the startup companies have status as "operating"

- Graphically representing the "category code" data: (Bar Chart)

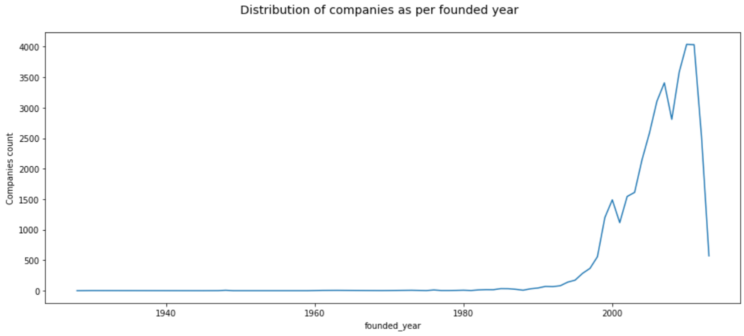


➢ Also, created a function for categorical keywords which is generally used for display boards and advertisements. In this, the font size of every word is based on frequency of company's top 10 category_code like software, mobile, etc.

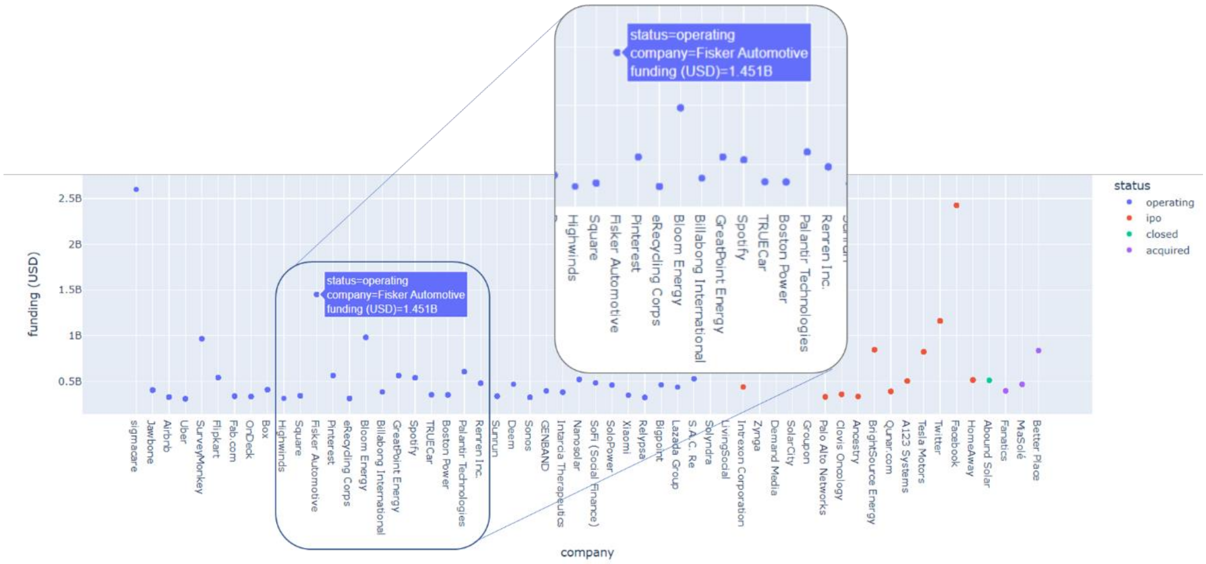


➢ **Data Transformation:**

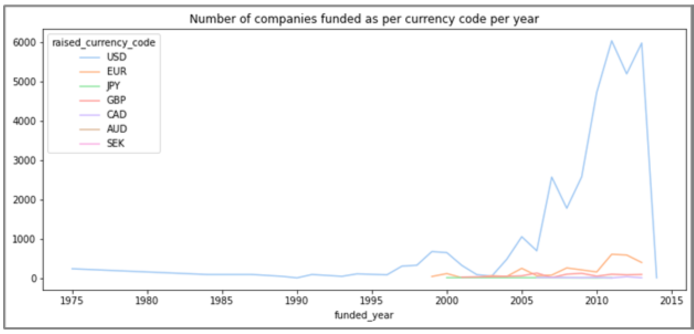
1. Converting object type (founded_at, funded_at) to datetime to analyse year-wise data



2. Changing strings to int type for interactive graphical representation & sorting. An interactive scatter plot is created for **companies** vs **funding USD amount** along with **current status**

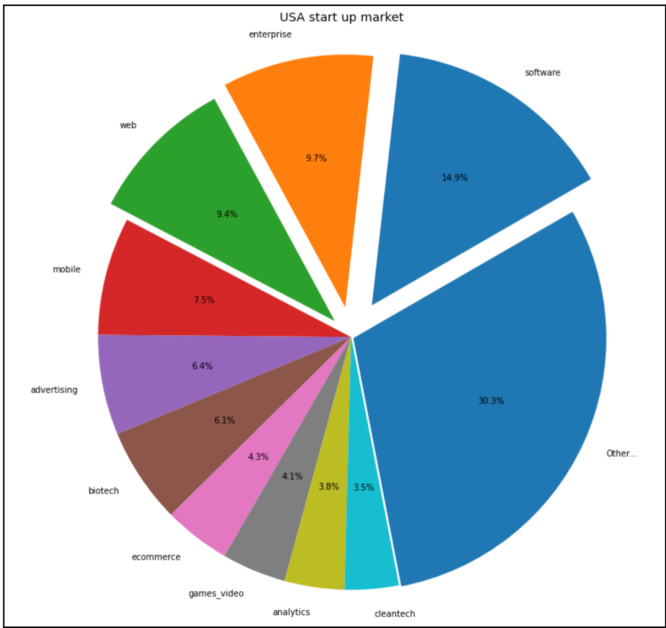


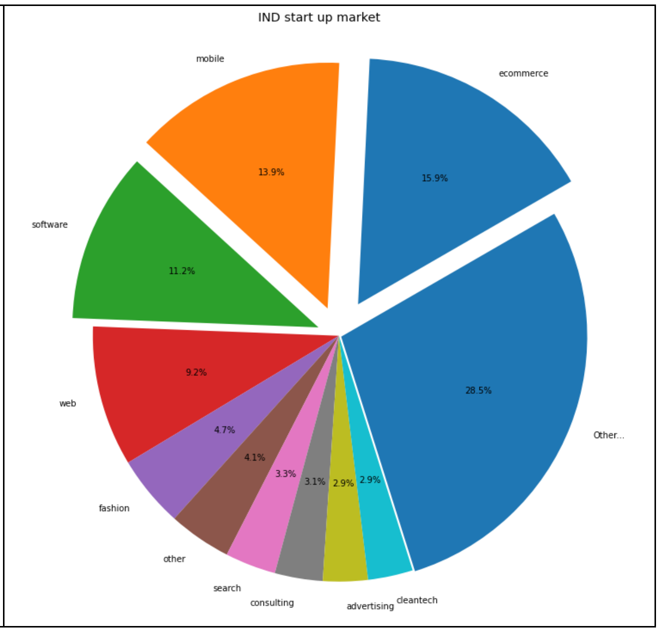
- Graphical representation of year wise funded currency code and corresponding raised amount:



index		raised_amount
0	AUD	40000000.0
1	CAD	6078000000.0
2	EUR	609298066933.0
3	GBP	37941000266.0
4	JPY	97800000025.0
5	SEK	80000000.0
6	USD	1482457799717.0

- Graphically presenting a pie chart for category of companies filtered for USA and INDIA:





For USA, most of start-up market is about Software, Enterprise and Web based companies
For India, most of start-up market is about ECommerce, Mobiles and Software based companies

- **Category grouping:**
  - Grouping categories together by their categorical status with new field category_group to build a correlation matrix as per domain knowledge. Few such category groups are mentioned below:

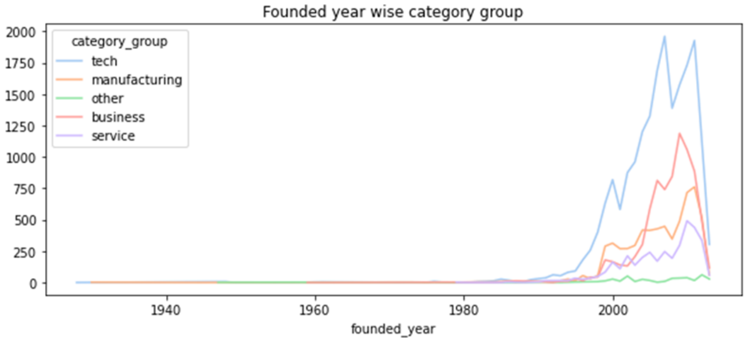
    Technical/IT/Computers related companies grouped under:

    tech: 'software', 'web', 'biotech', 'games_video', 'network_hosting', 'cleantech', 'nanotech', 'ecommerce', 'search', 'social', 'news', 'messaging', 'photo_video', 'music'

    Business related companies are grouped under:

    business: 'advertising', 'enterprise', 'consulting', 'analytics', 'ecommerce','finance', 'legal'

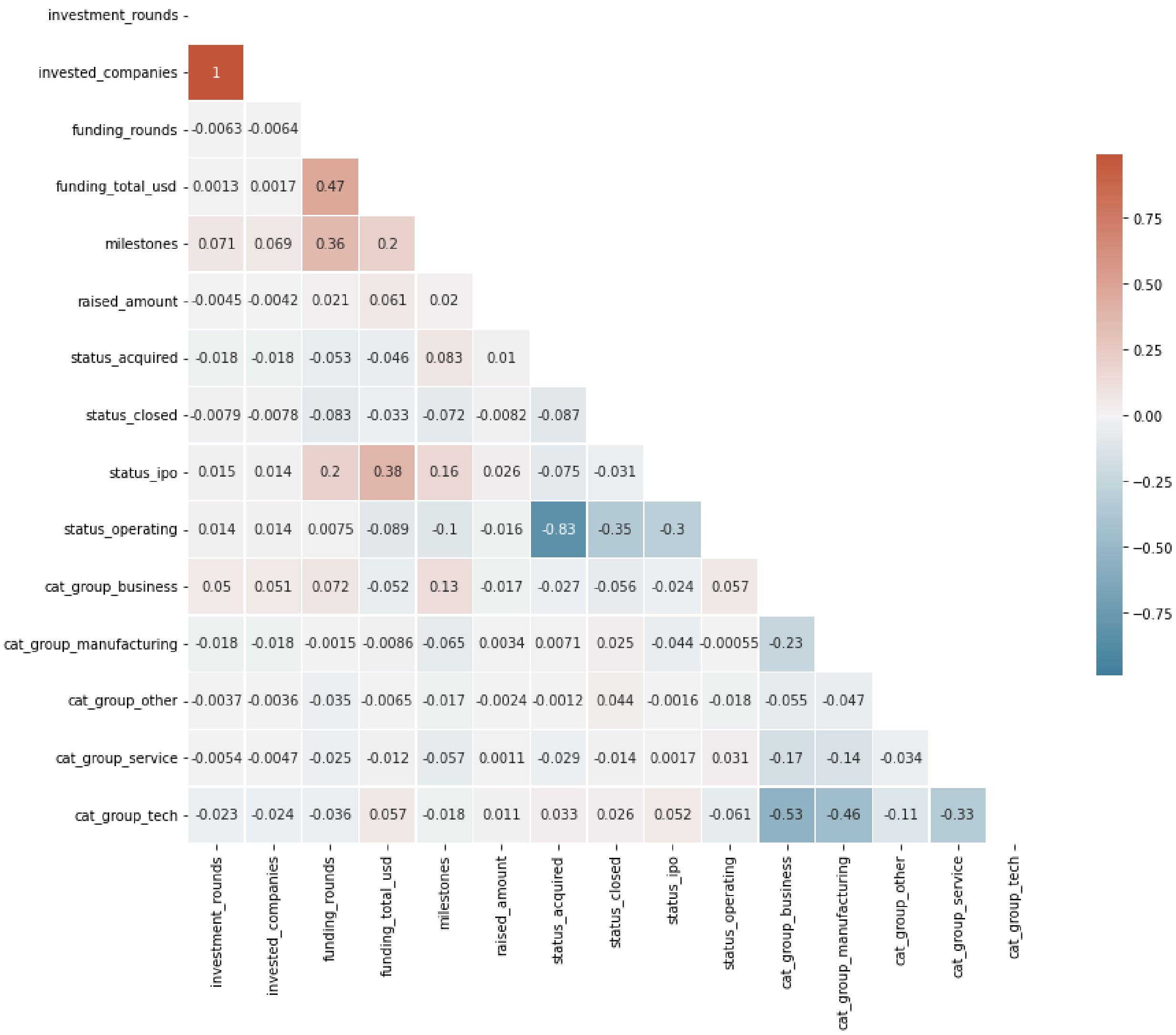
```
df_objects_investments_

tech             19463
business          8040
manufacturing     6329
service           3584
other              419
Name: category_group, d
```



- **Correlation matrix:**

  The correlation matrix shows the relationship of pairs of variables in the dataframe.

  - Coefficients close to 1 indicates a high positive correlation, which means that when one variable of the pair increases, so do the other one.
  - Coefficients close to -1 indicates a high negative correlation, meaning that when one variable of the pair increases, the other one decreases.
  - Coefficients close to 0 indicate no correlation between the pair, meaning the variables are independent of each other.



Some relations like invested_rounds/invested_companies (+ve), and status_operating / status acquired (-ve) are obvious and give no useful information. However, we can see how "status_ipo" is positively correlated to variables like "funding_total_usd"

➢ **Histogram:** Creating an interactive Histogram between Company Status and Investment Rounds

Pre-Processing Summary:

- Data quality issues are identified and addressed ✓
- Appropriate data pre-processing measures are applied wherever applicable ✓
- Any notable exceptions are reported in form of comments, wherever appropriate ✓
- Attempt in right direction to find out contributing factors: used correlation matrix ✓
- Right set of visuals are used for univariate and bivariate data analysis ✓
- Meaningful insights are derived and presented in effective manner ✓

Clustering

Variables (Columns) used for clustering 'category_group', 'status', 'funding_total_usd','country_code' as per the explained analysis

➢ **Label Encoding:**

Label encoding the data to convert categorical values into numerical values for clustering

	category_group	status	funding_total_usd	country_code
0	4	0	2347	65
1	4	0	2347	65
2	4	0	2347	65

➢ **MinMax Normalisation:**

In this technique of data normalization, linear transformation is performed on the original data with Min value = 0 and Max value = 1

```
[[  25    0 2347   65]
 [  25    0 2347   65]
 [  25    0 2347   65]
 ...
 [  36    3  387   19]
 [  14    3 1174   65]
 [   4    3 1281   65]]
```

```
[[0.625      0.         0.73046997 0.98484848]
 [0.625      0.         0.73046997 0.98484848]
 [0.625      0.         0.73046997 0.98484848]
 ...
 [0.9        1.         0.12044818 0.28787879]
 [0.35       1.         0.3653906  0.98484848]
 [0.1        1.         0.39869281 0.98484848]]
```

➢ **Dimensionality Reduction:**

Dimensionality reduction, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.

Principal Component Analysis (PCA):

The main linear technique for dimensionality reduction, principal component analysis, performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized.

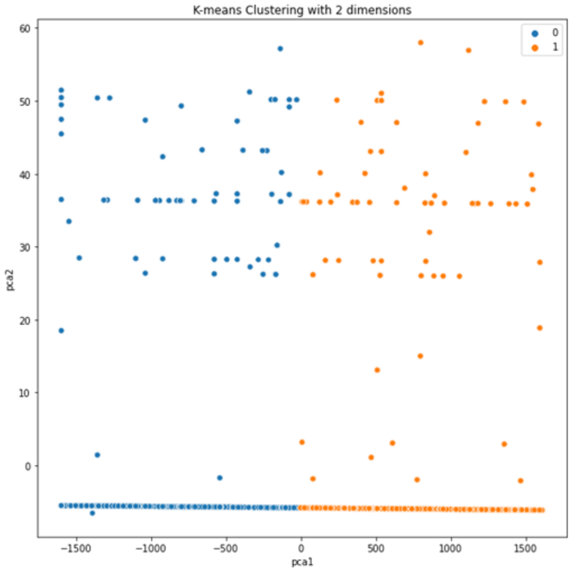
In this report, 4D (dimensional) data has been reduced to 2D (shown below) for 2D graphs and 3D data for 3D graph plotting.

```
[[0.625      0.         0.73046997 0.98484848]
 [0.625      0.         0.73046997 0.98484848]
 [0.625      0.         0.73046997 0.98484848]
 ...
 [0.9        1.         0.12044818 0.28787879]
 [0.35       1.         0.3653906  0.98484848]
 [0.1        1.         0.39869281 0.98484848]]
```

```
[[ 0.77283086  0.28864639]
 [ 0.77283086  0.28864639]
 [ 0.77283086  0.28864639]
 ...
 [-0.03236392 -0.60847362]
 [-0.22811598  0.02280939]
 [-0.33372867  0.24786074]]
```

- **Algorithms explored for clustering: using Calinski Harabasz (CH) Score**
  The Calinski-Harabasz index also known as the Variance Ratio Criterion, is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters, the higher the score, the better the performances.

1. **K-Medoids Clustering**
   K-Medoids is also called as Partitioning Around Medoid algorithm. K Medoids clustering algorithm that partitions sets of data points around a method (the least dissimilar point) and constantly attempts to lower the dissimilarity among the points in the same cluster. The key point here is that the medoid is a data point from input set, unlike in k means where mean is the mere average.

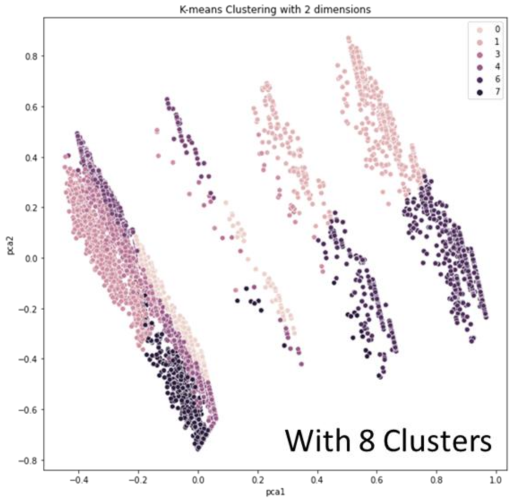



```
print("For 2 Clusters : \nCH

For 2 Clusters :
CH Score: 13399.365399326074
```

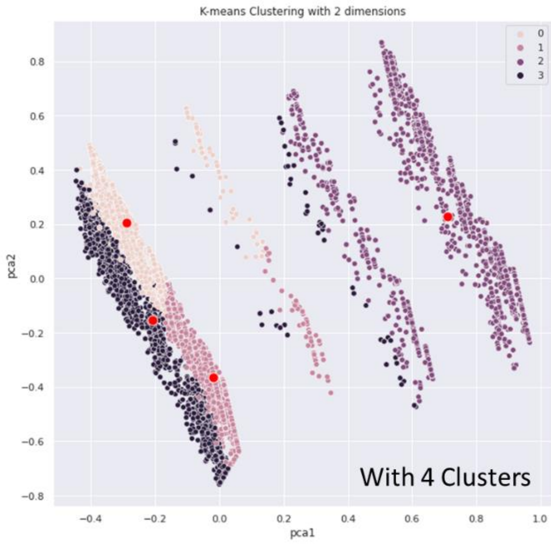
   Observations:
   - CH Score of 13,399 is observed. Higher this, score is better
   - PCA 2D graph is plotted since 4 dimensions are taken into account which require dimensionality reduction for graphical representation
   - Other algorithms are also explored for comparative analysis

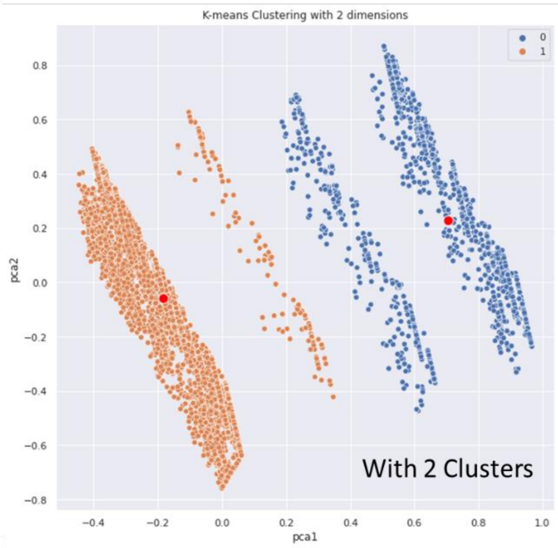
2. **K-Means Clustering**
   K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process.
   It is an iterative algorithm that divides the unlabelled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

```
3  Clusters :  2
   CH Score: 20017.188316999585
   Clusters :  3
   CH Score: 21524.198667569184
2  Clusters :  4
   CH Score: 21764.444417761402
   Clusters :  5
   CH Score: 21368.00340028302
   Clusters :  6
   CH Score: 21500.3087279139
   Clusters :  7
   CH Score: 21547.42242477534
1  Clusters :  8
   CH Score: 22481.79848924337
   Clusters :  9
   CH Score: 21547.17581813013
   Clusters :  10
   CH Score: 21572.081173096693
```

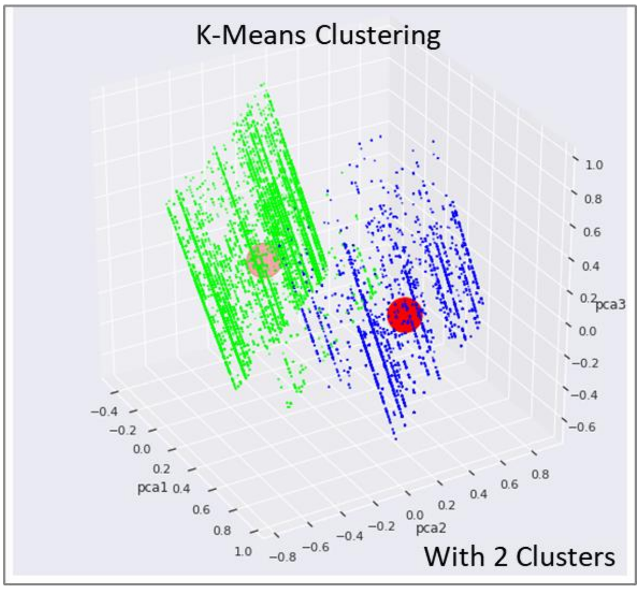


There is a lot of overlap among the clusters.
Since, in clustering the objective is to have clear distinction between the clusters, n_clusters = 4 and 2 has been explored as below





As seen from the graphs, n_clusters = 2 gives best distinctive clusters
3D PCA graph of the same is also plotted as below:

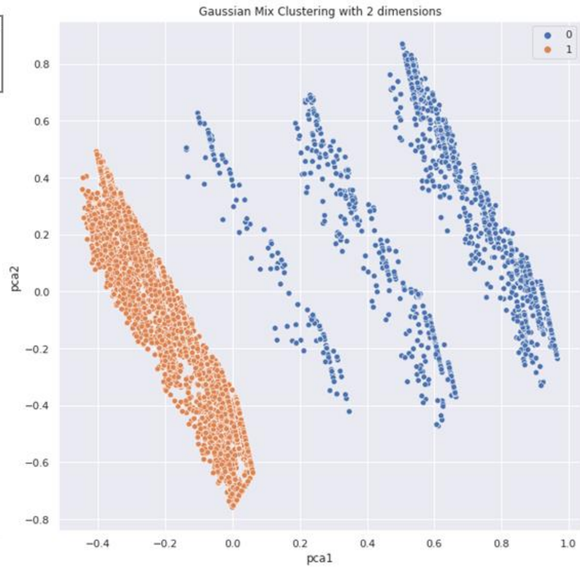


Logic behind the clustering is explored with the help of decision tree classification algorithm and is highlighted in the later section of the report

3. Gaussian Mix Clustering

GMM (or Gaussian Mixture Models) is an algorithm that uses the estimation of the density of the dataset to split the dataset in a preliminary defined number of clusters.

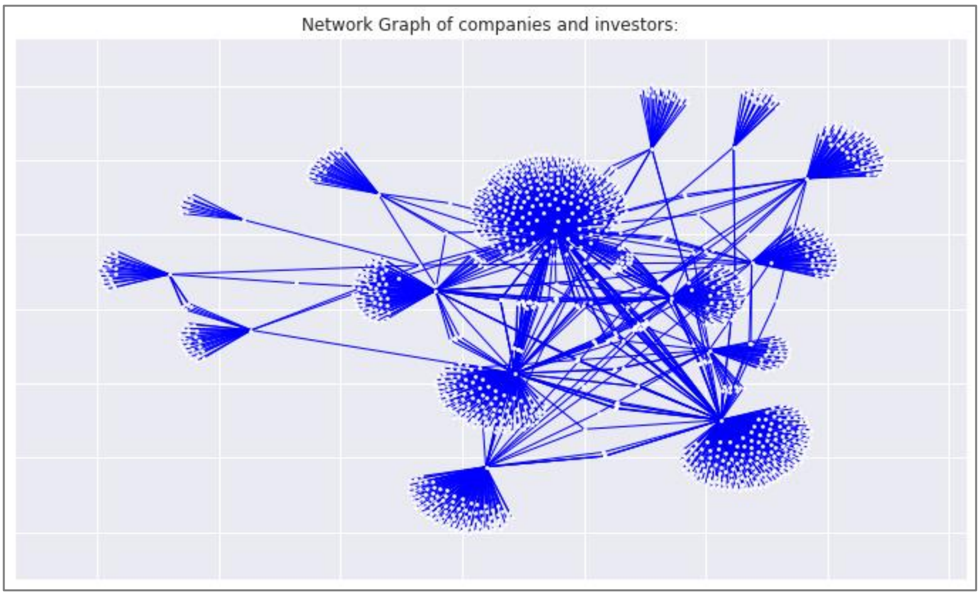
```
Clusters :  2
CH indicator: 18829.558486304788
Clusters :  3
CH indicator: 16731.537413046688
Clusters :  4
CH indicator: 11737.233146375082
Clusters :  5
CH indicator: 9760.135838147475
Clusters :  6
CH indicator: 12380.139822732015
Clusters :  7
CH indicator: 7748.433993905408
Clusters :  8
CH indicator: 11203.60091420432
Clusters :  9
CH indicator: 10404.152447724555
Clusters :  10
CH indicator: 10454.054145916361
```



As per Gaussian Mix Clustering algorithm, n_clusters = 2 shows best results similar to KMeans Clustering

- **Network Graph:**

Network Graph is a special representation of entities which have relationships among themselves. It is made up of a collection of two generic objects — (1) node: which represents an entity, and (2) edge: which represents the connection between any two nodes.



Network graph is plotted for first 5000 entries only to allow better visualization. From the network graph it is clear that, one fund may invest in a company which in turn may invest in other companies. These interconnections are clearly highlighted here.

Clustering Analysis

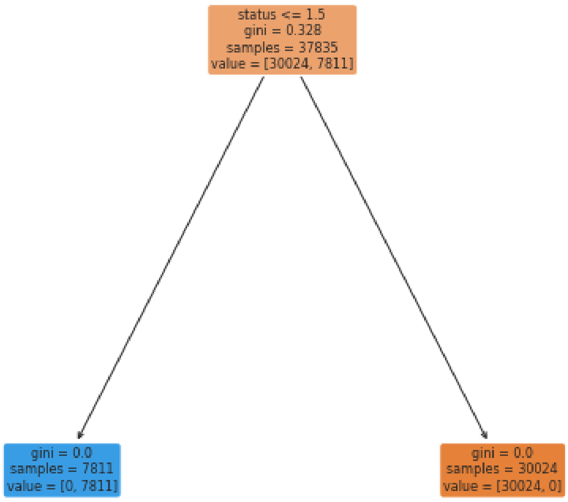
After clusters are formed using clustering algorithm, which is an unsupervised machine learning algorithm, it is essential to understand the logic behind clusters formation. To explore this, Decision Tree Classification algorithm has been deployed and results are explained below.

➢ **Decision Tree Classification:**

Data from K-Means with 2 clusters is selected for this study. Clusters so formed are taken as Labels for this supervised algorithm.
Dimensions / Columns: 'category_code', 'status', 'funding_total_ usd', 'country_code' are used.
Label encoding is used to make sure all columns/variables have numeric values. Classification ratio is observed to be :

```
Class=1, Count=7811, Percentage=20.645%
Class=0, Count=30024, Percentage=79.355%
```

This shows some bias towards Class 0.
Decision Tree from this data is plotted as below:



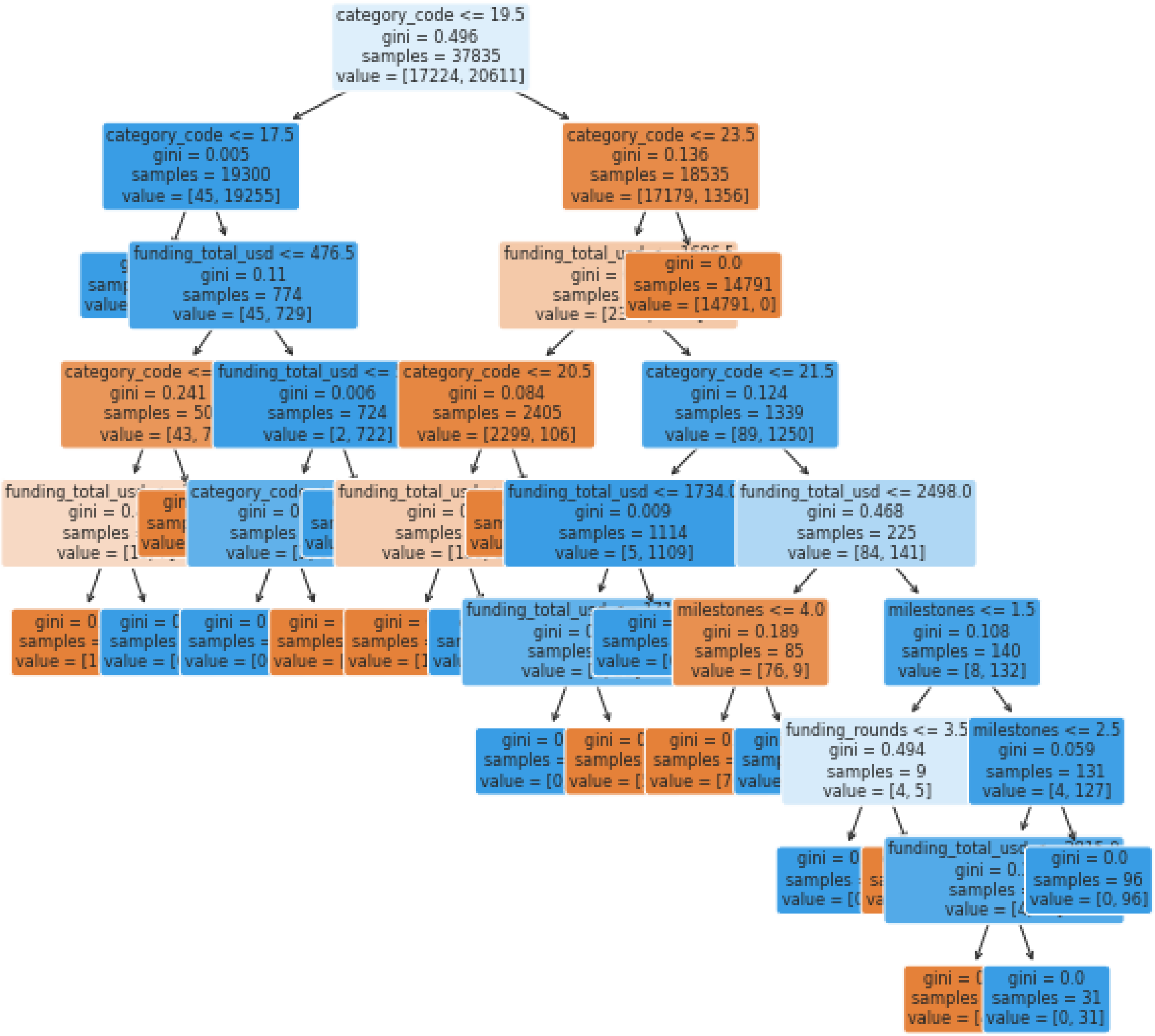
From the decision tree, it is clear that "Status" column is the reason behind this clustering. It is generally expected to take into account contribution of all columns/variables. This is explored further by removing the bias causing variable "status"

Dimensions / Columns: 'category_code', 'funding_total_ usd', 'country_code' are used.
Using the new data set in K-Means clustering and labelling as per its results, classification ratio is observed to be :

```
Class=0, Count=17224, Percentage=45.524%
Class=1, Count=20611, Percentage=54.476%
```

Much more uniform clustering is observed here.

Decision Tree is plotted for this:



The above Decision Tree shows contribution of all the variables selected for clustering. This is useful to understand the effect of each chosen variable and can help in making sound decisions for companies and investors.

Thank You