Brian Mallari - Unicorn Companies Exploratory Analyses - Python (pandas)

January 18, 2025

[27]:
```python
# Import pandas
import pandas as pd
```

[28]:
```python
# Read the 'Unicorn_Companies - Version for Python.csv' into a dataframe
df_copy_0 = pd.read_csv('Unicorn_Companies - Version for Python.csv')
```

[29]:
```python
# Look at the first few rows of this dataframe in order to confirm that things␣
↪are looking good so far
df_copy_0.head()
```

[29]:
```
      Id    Company Valuation  Valuation as Number Date Joined  \
0  10001  Bytedance     $180B         1.800000e+11    4/7/2017
1  10002     SpaceX     $100B         1.000000e+11   12/1/2012
2  10003      SHEIN     $100B         1.000000e+11    7/3/2018
3  10004     Stripe      $95B         9.500000e+10   1/23/2014
4  10005     Klarna      $46B         4.600000e+10  12/12/2011

                        Industry           City        Country  \
0        Artificial intelligence        Beijing          China
1                          Other      Hawthorne  United States
2  E-commerce & direct-to-consumer        Shenzhen          China
3                        Fintech  San Francisco  United States
4                        Fintech      Stockholm         Sweden

       Continent  Year Founded Funding  Funding as Number  \
0           Asia          2012     $8B       8.000000e+09
1  North America          2002     $7B       7.000000e+09
2           Asia          2008     $2B       2.000000e+09
3  North America          2010     $2B       2.000000e+09
4         Europe          2005     $4B       4.000000e+09

                                 Select Investors
0  Sequoia Capital China, SIG Asia Investments, S…
1  Founders Fund, Draper Fisher Jurvetson, Rothen…
2  Tiger Global Management, Sequoia Capital China…
3        Khosla Ventures, LowercaseCapital, capitalG
```

```
4   Institutional Venture Partners, Sequoia Capita…
```

[30]:
```
df_copy_0.dtypes
```

[30]:
```
Id                        int64
Company                  object
Valuation                object
Valuation as Number     float64
Date Joined              object
Industry                 object
City                     object
Country                  object
Continent                object
Year Founded              int64
Funding                  object
Funding as Number       float64
Select Investors         object
dtype: object
```

1 Which unicorn companies have had the biggest return on investment?

[32]:
```
# Retain only the rows where 'Funding as Number' is greater than zero (in order␣
 ↪for the math to work out correctly)
df_copy_1a = df_copy_0[
    0 < df_copy_0['Funding as Number']
]

# Retain only Company, Valuation as Number, and Funding as Number
df_copy_1b = df_copy_1a.loc[:, ['Company', 'Valuation as Number', 'Funding as␣
 ↪Number']]

# Add a new column to retain the ROI, rounded to two decimal places
df_copy_1c = df_copy_1b
df_copy_1c['ROI'] = round((df_copy_1c['Valuation as Number'] -␣
 ↪df_copy_1b['Funding as Number'])/df_copy_1b['Funding as Number'], 2)

# Sort the ROI values from highest to lowest and showcase only the top 10␣
 ↪companies by ROI
df_copy_1d = df_copy_1c.sort_values(by = ['ROI'], ascending = False).head(10)

# Blank out the indecies in order to make the output look neater
black_indeces = [''] * len(df_copy_1d)
df_copy_1final = df_copy_1d
df_copy_1final.index = black_indeces
df_copy_1final
```

[32]:
```
                   Company  Valuation as Number  Funding as Number      ROI
                    Zapier         4.000000e+09          1000000.0  3999.00
                    Dunamu         9.000000e+09         71000000.0   125.76
                  Workhuman        1.000000e+09          9000000.0   110.11
                      CFGI         2.000000e+09         19000000.0   104.26
                    Manner         1.000000e+09         10000000.0    99.00
            DJI Innovations        8.000000e+09        105000000.0    75.19
                GalaxySpace        1.000000e+09         14000000.0    70.43
                      Canva        4.000000e+10        572000000.0    68.93
                 Il Makiage        2.000000e+09         29000000.0    67.97
      Revolution Precrafted        1.000000e+09         15000000.0    65.67
```

2 How long does it usually take for a company to become a unicorn? Has it always been this way?

According to the original file, the "join year" for the company Yidian Zixun is BEFORE the founding year, and this doesn't make sense because a company should have been founded FIRST before establishing any valuation. However, according to Crunchbase, Yidian Zixun was founded in 2013, and the last round of funding for the company was on November 14, 2017 as part of a Series E round (source = https://www.crunchbase.com/organization/yidian-zixun). There may have been an error when recording data, or there may have been some technicality that would've placed the company's "join year" before the founding year, like a corporate restructuring. However, in order for the math to make sense, this company was excluded from the aggreggation.

[35]:
```python
# Retain the companies that aren't 'Yidian Zixun
df_copy_2_part1_a = df_copy_0[df_copy_0['Company'] != 'Yidian Zixun']

# Retain only the Company, Date Joined, and Year Founded
df_copy_2_part1_b = df_copy_2_part1_a[['Company', 'Date Joined', 'Year␣
↪Founded']]

# Copy the above dataframe and then format the date to use dashes instead of␣
↪forwards slashes
df_copy_2_part1_c = df_copy_2_part1_b.loc[:,:]
df_copy_2_part1_c['Date Joined'] = pd.to_datetime(df_copy_2_part1_c['Date␣
↪Joined'], format = '%m/%d/%Y')

# Copy the above dataframe and then add a new column for just the year that the␣
↪company had joined the list of unicorn companies
df_copy_2_part1_d = df_copy_2_part1_c.loc[:,:]
df_copy_2_part1_d['Year Joined'] = df_copy_2_part1_d['Date Joined'].dt.year

# Copy the above dataframe and then add a new column for the number of years␣
↪elapsed from founding to gaining unicorn status
df_copy_2_part1_e = df_copy_2_part1_d.loc[:,:]
```

```
df_copy_2_part1_e['Years to Unicorn Status'] = (df_copy_2_part1_e['Year␣
↪Joined'] - df_copy_2_part1_e['Year Founded'])

# Find the average of years until a company had achieved unicorn status
avg_yrs_to_unicorn_status_for_all_years = round(df_copy_2_part1_e['Years to␣
↪Unicorn Status'].mean(), 2)

# Create a new dataframe to hold this calculated data
pending_data = {'Avg. Years to Unicorn Status for All Years':␣
↪[avg_yrs_to_unicorn_status_for_all_years]}
df_copy_2_part1_f = pd.DataFrame(pending_data)
df_copy_2_part1_f

# Blank out the index for the sake of cleaning up the output
df_copy_2_part1_final = df_copy_2_part1_f.loc[:,:]
blank_indeces = [''] * len(df_copy_2_part1_f)
df_copy_2_part1_final.index = blank_indeces
df_copy_2_part1_final
```

[35]:

```
  Avg. Years to Unicorn Status for All Years
                                        7.01
```

[36]:

```
# Copy the dataframe above that includes years to unicorn status
df_copy_2_part2_a = df_copy_2_part1_e

# Group the rows by year founded and then take the averege of the years to␣
↪unicorn status for each group
df_copy_2_part2_b = df_copy_2_part2_a.groupby('Year Founded', as_index =␣
↪False)['Years to Unicorn Status'].mean()

# Explicitly sort the rows by year founded in ascending order just to be safe
df_copy_2_part2_c = df_copy_2_part2_b.sort_values('Year Founded')

# Rename the column with the average years to unicorn status for better␣
↪understandability
df_copy_2_part2_d = df_copy_2_part2_c.rename(columns = {'Years to Unicorn␣
↪Status' : 'Avg. Years to Unicorn Status'})

# Blank out the index for the sake of cleaning up the output
df_copy_2_part2_final = df_copy_2_part2_d.loc[:,:]
blank_indeces = [''] * len(df_copy_2_part2_d)
df_copy_2_part2_final.index = blank_indeces
df_copy_2_part2_final
```

[36]:

```
  Year Founded  Avg. Years to Unicorn Status
          1919                     98.000000
          1979                     37.000000
```

```
1984                37.000000
1990                27.000000
1991                27.000000
1992                25.000000
1993                28.000000
1994                21.500000
1995                21.500000
1996                25.000000
1997                20.000000
1998                19.800000
1999                19.375000
2000                19.272727
2001                16.666667
2002                13.250000
2003                15.500000
2004                15.250000
2005                13.571429
2006                12.933333
2007                11.500000
2008                11.222222
2009                10.147059
2010                 8.950000
2011                 8.353659
2012                 7.463158
2013                 6.712644
2014                 6.018349
2015                 5.200000
2016                 4.463636
2017                 3.810811
2018                 3.114754
2019                 2.177778
2020                 1.080000
2021                 0.400000
```

3 Which countries have the most unicorns? Are there any cities that appear to be industry hubs?

[38]:
```
# Copy counts of industries grouped by industry and convert the output to a␣
 ↪dataframe
df_copy_3_part1_a = df_copy_0.groupby('Industry')['Industry'].count().to_frame()

# Rename that one column for clarity and to prepare for the conversion of the␣
 ↪index into its own column
# (both the column and index ended up with the same name after the above line␣
 ↪of code)
df_copy_3_part1_b = df_copy_3_part1_a.rename(columns = {'Industry': 'Count'})
```

```
# Convert the index to its own column for improved aesthetics
df_copy_3_part1_c = df_copy_3_part1_b.reset_index()

# Sort the rows in descending order by count
df_copy_3_part1_d = df_copy_3_part1_c.sort_values('Count', ascending = False)

# Limit the output to just ten rows
df_copy_3_part1_e = df_copy_3_part1_d.head(10)

# Blank out the index for the sake of a cleaner output
df_copy_3_part1_final = df_copy_3_part1_e
blank_indeces = [''] * len(df_copy_3_part1_e)
df_copy_3_part1_final.index = blank_indeces
df_copy_3_part1_final
```

[38]:

```
                             Industry  Count
                              Fintech    224
         Internet software & services    205
      E-commerce & direct-to-consumer    111
                               Health     74
              Artificial intelligence     73
                                Other     58
   Supply chain, logistics, & delivery    57
                        Cybersecurity     50
          Data management & analytics     41
          Mobile & telecommunications     38
```

[39]:

```
# Copy counts of industries grouped first by city then industry and then␣
↪convert the output to a dataframe
df_copy_3_part2_a = df_copy_0.groupby(['City', 'Industry'])['Industry'].count().
↪to_frame()

# Rename that one column for clarity and to prepare for the conversion of the␣
↪index into its own column
# (both column and index ended up with the same after the above line of code)
df_copy_3_part2_b = df_copy_3_part2_a.rename(columns = {'Industry': 'Count'})

# Convert the indeces to their own column for improved aesthetics
df_copy_3_part2_c = df_copy_3_part2_b.reset_index()

# Sort the rows in in descending order by count
df_copy_3_part2_d = df_copy_3_part2_c.sort_values('Count', ascending = False)

# Limit the output to just ten rows
df_copy_3_part2_e = df_copy_3_part2_d.head(10)
```

```
# Blank out the index for the sake of a cleaner output
df_copy_3_part2_final = df_copy_3_part2_e
blank_indeces = [''] * len(df_copy_3_part2_e)
df_copy_3_part2_final.index = blank_indeces
df_copy_3_part2_final
```

[39]:

```
           City                        Industry  Count
  San Francisco     Internet software & services     54
  San Francisco                         Fintech     41
       New York                         Fintech     33
         London                         Fintech     24
       New York     Internet software & services     20
       New York                          Health     14
  San Francisco                          Health     12
        Beijing  E-commerce & direct-to-consumer     11
       Shanghai            Auto & transportation     10
      Bengaluru     Internet software & services      9
```

4 Which investors have funded the most unicorns?

[41]:

```
# Split the Select Investors column on commas into separate columns
df_copy_4a = df_copy_0['Select Investors'].str.split(',', expand = True)

# Stack the newly created columns into a single column and convert the output␣
 ↪into a dataframe
df_copy_4b = df_copy_4a.stack().to_frame()

# Eliminate the rows with None as values just in case they hadn't been␣
 ↪automatically removed already by this point
df_copy_4c = df_copy_4b.dropna()

# Strip away any whitespaces around the words and convert the output into a␣
 ↪dataframe
df_copy_4d = df_copy_4c[0].str.strip().to_frame()

# Group the rows by investor, output the count of each group, and convert the␣
 ↪output to a dataframe
df_copy_4e = df_copy_4d.groupby(0)[0].count().to_frame()

# Rename the column for clarity and to prepare for the conversion of the index␣
 ↪into its own column
# (both column and index ended up with the same after the above line of code)
df_copy_4f = df_copy_4e.rename(columns = {0: 'Count'})

# Convert the indeces to their own column for improved aesthetics
df_copy_4g = df_copy_4f.reset_index()
```

```
# Rename the investor column for improved understandability
df_copy_4h = df_copy_4g.rename(columns = {0: 'Investor'})

# Sort the rows in descening order by count
df_copy_4i = df_copy_4h.sort_values('Count', ascending = False)

# Limit the output to just ten rows
df_copy_4j = df_copy_4i.head(10)

# Blank out the index for the sake of a cleaner output
df_copy_4final = df_copy_4j
blank_indeces = [''] * len(df_copy_4j)
df_copy_4final.index = blank_indeces
df_copy_4final
```

[41]:

```
                     Investor  Count
                        Accel     60
     Tiger Global Management     53
         Andreessen Horowitz     53
        Sequoia Capital China     48
              Sequoia Capital     47
             Insight Partners     47
             General Catalyst     34
  Lightspeed Venture Partners     34
                SoftBank Group     34
               Index Ventures     32
```