Download Crunchbase Datasets

source: GitHub (notpeter/crunchbase-data)

release: 2015-08-27

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
from difflib import SequenceMatcher
import math

companies = pd.read_csv('https://raw.githubusercontent.com/notpeter/crunchbase-data/master/companies.csv')
investments = pd.read_csv('https://raw.githubusercontent.com/notpeter/crunchbase-data/master/investments.csv')
acquisitions = pd.read_csv('https://raw.githubusercontent.com/notpeter/crunchbase-data/master/acquisitions.csv')
rounds = pd.read_csv('https://raw.githubusercontent.com/notpeter/crunchbase-data/master/rounds.csv')
additions = pd.read_csv('https://raw.githubusercontent.com/notpeter/crunchbase-data/master/additions.csv')
```

https://www.kaggle.com/kbrookshier/crunchbase-startup-investments

Data Transformation

Required New Variables

Name	Description	Sparsity Level (Avg=69%)	Average
roundD	The company did a round D	98.8%	
roundC	The company did a round C	95.9%	
roundB	The company did a round B	92.5%	
roundA	The company did a round A	88.8%	
VentureCapital	Has venture capital (with missing values)	60.7%	
isTech	Is a tech company	0.0%	
target	The company was acquired by other or went to a public stock market (IPO)	0.0%	
roundD_raised_amount	Raised amount of Round D	98.8%	$40.449.855
roundC_raised_amount	Raised amount of Round C	96.0%	$21.162.205
roundB_raised_amount	Raised amount of Round B	92.9%	$14.968.688
roundA_raised_amount	Raised amount of Round A	89.7%	$7.640.412
investment_per_round	Total US Dollars invested per round of investment	61.5%	$9.161.589
funding_total_usd	Total funding in US dollars	61.5%	$21.646.508
roundD_age	Company's age when it did its round D	98.8%	6,5
total_investments	Total number of investments made by the company	98.5%	2,7
customer_count	Number of customers	98.3%	6,0
ipo_age	Company's age when it went to a public stock market	96.4%	7,7
roundC_age	Company's age when it did its round C	95.9%	5,4
total_acquisitions	Number of acquisitions made by the company	93.9%	2,2
competitor_acquired_ipo	Number of competitors either acquired or IPO'd	93.1%	2,0
roundB_age	Company's age when it did its round B	92.5%	4,1
roundA_age	Company's age when it did its round A	88.7%	3,3
competitor_count	Number of competitors	87.8%	3,2
age_Acquired	Company's age when acquired	87.1%	9,2
success_age	Age of company when it got acquired or went to a public stock market (IPO)	84.3%	8,6
top500_investor	Number of top500 investors in the company (Top 500 by number of investments made by investor)	81.2%	3,5
investors_per_round	Number of investors per round	70.0%	2,3
total_exp_founders_years	Total experience of founders in years	70.0%	9,5
age_first_funding_year	Company's age when it received first funding	53.9%	3,2
funding_rounds	Number of funding rounds	53.9%	2,1
totalFounders	Number of founders	41.4%	1,7
total_experience_jobs_years	Total experience of total jobs in the company in years	28.8%	12,0
totaljobs	Total jobs of the company	23.9%	5,3
age_yrs	Actual age in years	0.0%	10,3

Not-Age Variables

All except age variables - Total count: 21

```
df = pd.DataFrame(data=companies[['permalink','name']].values, columns = ['permalink', 'name'])
df
```

	permalink	name
0	/organization/-fame	#fame
1	/organization/-qounter	:Qounter
2	/organization/-the-one-of-them-inc-	(THE) ONE of THEM,Inc.
3	/organization/0-6-com	0-6.com
4	/organization/004-technologies	004 Technologies
...	...	...
66363	/organization/zznode-science-and-technology-co...	ZZNode Science and Technology
66364	/organization/zzzzapp-com	Zzzzapp Wireless ltd.
66365	/organization/Áeron	ÁERON
66366	/organization/Ôasys-2	Ôasys
66367	/organization/İnovatiff-reklam-ve-tanıtım-hizm...	İnovatiff Reklam ve Tanıtım Hizmetleri Tic

66368 rows × 2 columns

```
#roundD - 1 if company did a round-D else 0
df['roundD'] = np.zeros(df.shape[0])

#roundC - 1 if company did a round-C else 0
df['roundC'] = np.zeros(df.shape[0])
```

```
#roundB - 1 if company did a round-B else 0
df['roundB'] = np.zeros(df.shape[0])

#roundA - 1 if company did a round-A else 0
df['roundA'] = np.zeros(df.shape[0])

#VentureCapital = 1 if company has a venture else 0
df['VentureCapital'] = np.zeros(df.shape[0])

#IsTech = 1 if company is a Tech company else 0
df['IsTech'] = np.empty(df.shape[0])
df['IsTech'] = np.nan
tech_keys = ['tech','analytics', 'software', 'elec', 'web','manufacturing','internet','auto','smart','e-','data','develop','product',

#target = 1 if company went into acquisition (or IPO) else 0
df['target'] = np.zeros(df.shape[0])

#roundD_raised_amount = total amount raised by company in D rounds
df['roundD_raised_amount'] = np.empty(df.shape[0])
df['roundD_raised_amount'][:] = np.nan

#roundC_raised_amount = amount raised by company in C rounds
df['roundC_raised_amount'] = np.empty(df.shape[0])
df['roundC_raised_amount'][:] = np.nan

#roundB_raised_amount = total amount raised by company in B rounds
df['roundB_raised_amount'] = np.empty(df.shape[0])
df['roundB_raised_amount'][:] = np.nan

#roundA_raised_amount = total amount raised by company in A rounds
df['roundA_raised_amount'] = np.empty(df.shape[0])
df['roundA_raised_amount'][:] = np.nan

#total_investments = total no. of investments made to the company
df['total_investments'] = np.empty(df.shape[0])
df['total_investments'][:] = np.nan
```

```
#investment_per_round = average fund raised per round
df['investment_per_round'] = np.empty(df.shape[0])
df['investment_per_round'][:] = np.nan

#funding_total_usd = total amound raised by company in all rounds
df['funding_total_usd'] = np.empty(df.shape[0])
df['funding_total_usd'][:] = np.nan

#total_acquisitions = no. of times company went into acquisition (or IPO)
df['total_acquisitions'] = np.zeros(df.shape[0])

#competitors_count = no. of competitiors
df['competitors_count'] = np.empty(df.shape[0])
df['competitors_count'][:] = np.nan

#competitors_acquired = no. of competitors went into acquisition (or IPO)
df['competitors_acquired'] = np.empty(df.shape[0])
df['competitors_acquired'][:] = np.nan

#country_code = Country Code (if available)
df['country_code'] = np.empty(df.shape[0])
df['country_code'][:] = np.nan

#funding_rounds = No of funding rounds company went in
df['funding_rounds'] = np.empty(df.shape[0])
df['funding_rounds'][:] = np.nan

#investors_per_round = average no. of investors per round
df['investors_per_round'] = np.empty(df.shape[0])
df['investors_per_round'][:] = np.nan

#top500_investors = average no. of investors per round
df['top500_investors'] = np.empty(df.shape[0])
df['top500_investors'][:] = np.nan
top_investors = investments['investor_name'].value_counts()[:500].index.tolist()

for i in range(30800,31900):
  cp = df.loc[i,'permalink']
```

```
    cp = df.loc[i,'permalink']
    print("Companies completed:", i, " ,out of", companies.shape[0])

    company_details = companies.loc[companies['permalink']==cp]
    investments_in_company = investments.loc[investments['company_permalink'] ==  cp]
    company_acquisitions = acquisitions.loc[acquisitions['company_permalink'] == cp]

    #Company Details:

    company_cat = company_details.loc[i,'category_list']
    if (isinstance(company_cat,str)):
      ## isTech
      df.loc[i,'IsTech'] = 0
      for key in tech_keys:
        if key in company_cat.lower():
          df.loc[i,'IsTech'] = 1
          break


    ## funding_rounds
    df.loc[i,'funding_rounds'] = company_details.loc[i,'funding_rounds']

    try:
      ## investment_per_round
      df.loc[i,'investment_per_round'] = float(company_details.loc[i,'funding_total_usd'])/float(company_details.loc[i,'funding_rounds

      ## funding_total_usd
      df.loc[i,'funding_total_usd'] = company_details.loc[i,'funding_total_usd']
    except:
      df.loc[i,'investment_per_round'] = np.nan
      df.loc[i,'funding_total_usd'] = np.nan

    ## investor_per_round - will be replaced from data in Investments Details (if available)
    df.loc[i, 'investors_per_round'] = 2

    ## total_investments - will be replaced from data in Investments Details (if available)
    df.loc[i,'total_investments'] = float(company_details.loc[i,'funding_rounds'])*2
```

```
## country_code
df.loc[i,'country_code'] = company_details.loc[i,'country_code']


#Investment Details:

if investments_in_company.shape[0] != 0:

  if 'D' in investments_in_company['funding_round_code'].tolist():
    ## roundD
    df.loc[i,'roundD'] = 1
    amt_list = investments_in_company.loc[investments_in_company['funding_round_code']=='D']['raised_amount_usd']
    amt_list.dropna()
    ## roundD_raised_amound
    df.loc[i,'roundD_raised_amount'] = np.mean(amt_list)

  if 'C' in investments_in_company['funding_round_code'].tolist():
    ## roundC
    df.loc[i,'roundC'] = 1
    amt_list = investments_in_company.loc[investments_in_company['funding_round_code']=='C']['raised_amount_usd']
    amt_list.dropna()
    ## roundC_raised_amound
    df.loc[i,'roundC_raised_amount'] = np.mean(amt_list)

  if 'B' in investments_in_company['funding_round_code'].tolist():
    ## roundB
    df.loc[i,'roundB'] = 1
    amt_list = investments_in_company.loc[investments_in_company['funding_round_code']=='B']['raised_amount_usd']
    amt_list.dropna()
    ## roundB_raised_amound
    df.loc[i,'roundB_raised_amount'] = np.mean(amt_list)

  if 'A' in investments_in_company['funding_round_code'].tolist():
    ## roundA
    df.loc[i,'roundA'] = 1
    amt_list = investments_in_company.loc[investments_in_company['funding_round_code']=='A']['raised_amount_usd']
    amt_list.dropna()
    ## roundA_raised_amount
```

```
    df.loc[i,'roundA_raised_amount'] = np.mean(amt_list)

  ## VentureCapital
  if 'venture' in investments_in_company['funding_round_type'].tolist():
    df.loc[i,'VentureCapital'] = 1

  ## total_investments
  df.loc[i,'total_investments'] = investments_in_company.shape[0]

  ## investor_per_round
  df.loc[i, 'investors_per_round'] = investments_in_company.shape[0]/company_details.loc[i, 'funding_rounds']

  ## top500_investors
  df.loc[i,'top500_investors'] = 0
  for investor in investments_in_company['investor_name'].tolist():
      if investor in top_investors:
        df.loc[i,'top500_investors'] += 1

#Acquisition Details:

if company_acquisitions.shape[0] != 0:
  ## target
  df.loc[i,'target'] = 1
  ## total_acquisitions
  df.loc[i,'total_acquisitions'] = company_acquisitions.shape[0]


#Competition Details

competitors = companies.loc[companies['permalink']!=cp]
if (isinstance(company_cat,str)):
   df.loc[i,'competitors_count'] = 0
   df.loc[i,'competitors_acquired'] = 0
   for j in competitors.index:
    try:
      if SequenceMatcher(None,competitors.loc[j,'category_list'],company_cat).ratio()>0.7:
        ## competitors_count
        df.loc[i,'competitors_count'] += 1
```

```
        df.loc[i,'competitors_count'] += 1
        if competitors.loc[j,'permalink'] in acquisitions['company_permalink'].tolist():
          ## competitors_acquired
          df.loc[i,'competitors_acquired'] += 1
    except:
      continue

df.to_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_7a.csv', index=False)
```

Age Variables

```
class Date:
    def __init__(self, d, m, y):
        self.d = d
        self.m = m
        self.y = y
monthDays = [31, 28, 31, 30, 31, 30,
             31, 31, 30, 31, 30, 31]
def countLeapYears(d):
    years = d.y

    if (d.m <= 2):
        years -= 1
    ans = int(years / 4)
    ans -= int(years / 100)
    ans += int(years / 400)
    return ans
def getDifference(dt1, dt2):
    n1 = dt1.y * 365 + dt1.d
    for i in range(0, dt1.m - 1):
        n1 += monthDays[i]
    n1 += countLeapYears(dt1)

    n2 = dt2.y * 365 + dt2.d
    for i in range(0, dt2.m - 1):
```

```
    for i in range(0, dt2.m - 1):
        n2 += monthDays[i]
    n2 += countLeapYears(dt2)

    return (n2 - n1)
```

```
## roundX_age -  Company's age when it did its round X
df_4 = pd.DataFrame(data=investments.values)
roundA_age = pd.DataFrame(columns = ['permalink', 'name', 'age'])
roundB_age = pd.DataFrame(columns = ['permalink', 'name', 'age'])
roundC_age = pd.DataFrame(columns = ['permalink', 'name', 'age'])
roundD_age = pd.DataFrame(columns = ['permalink', 'name', 'age'])
df_4 = df_4[[0,1,15,16]]
for i in range(df_4.shape[0]):
  if df_4.iloc[i][15]=='A':
    start = df_4.iloc[i][1]
    j,k = (df_3.applymap(lambda x: str(x).startswith(start))).values.nonzero()
    start_date = df_3.iloc[j[0]][11]
    end_date = df_4.iloc[i][16]
    if type(start_date) == float or type(end_date) == float:
      continue
    st_dt = [int(it) for it in start_date.replace('-', ' ').split(' ')]
    ed_dt = [int(it) for it in end_date.replace('-', ' ').split(' ')]
    roundA_age = roundD_age.append({'permalink' : df_4.iloc[i][0], 'name' : df_4.iloc[i][1], 'age' : getDifference(Date(st_dt[2],st_
            ignore_index = True)
  if df_4.iloc[i][15]=='B':
    start = df_4.iloc[i][1]
    j,k = (df_3.applymap(lambda x: str(x).startswith(start))).values.nonzero()
    start_date = df_3.iloc[j[0]][11]
    end_date = df_4.iloc[i][16]
    if type(start_date) == float or type(end_date) == float:
      continue
    st_dt = [int(it) for it in start_date.replace('-', ' ').split(' ')]
    ed_dt = [int(it) for it in end_date.replace('-', ' ').split(' ')]
    roundB_age = roundD_age.append({'permalink' : df_4.iloc[i][0], 'name' : df_4.iloc[i][1], 'age' : getDifference(Date(st_dt[2],st_
            ignore_index = True)
  if df_4.iloc[i][15]=='C':
    start = df_4.iloc[i][1]
```

```
    start = df_4.iloc[i][1]
    j,k = (df_3.applymap(lambda x: str(x).startswith(start))).values.nonzero()
    start_date = df_3.iloc[j[0]][11]
    end_date = df_4.iloc[i][16]
    if type(start_date) == float or type(end_date) == float:
      continue
    st_dt = [int(it) for it in start_date.replace('-', ' ').split(' ')]
    ed_dt = [int(it) for it in end_date.replace('-', ' ').split(' ')]
    roundC_age = roundD_age.append({'permalink' : df_4.iloc[i][0], 'name' : df_4.iloc[i][1], 'age' : getDifference(Date(st_dt[2],st_
                ignore_index = True)
  if df_4.iloc[i][15]=='D':
    start = df_4.iloc[i][1]
    j,k = (df_3.applymap(lambda x: str(x).startswith(start))).values.nonzero()
    start_date = df_3.iloc[j[0]][11]
    end_date = df_4.iloc[i][16]
    if type(start_date) == float or type(end_date) == float:
      continue
    st_dt = [int(it) for it in start_date.replace('-', ' ').split(' ')]
    ed_dt = [int(it) for it in end_date.replace('-', ' ').split(' ')]
    roundD_age = roundD_age.append({'permalink' : df_4.iloc[i][0], 'name' : df_4.iloc[i][1], 'age' : getDifference(Date(st_dt[2],st_
                ignore_index = True)
```

```
## age_yrs
df_4 = pd.DataFrame(data=companies.values)
actual_age = pd.DataFrame(columns = ['permalink', 'name', 'age'])

df_4 = df_4[[0,1,5,11]]
for i in range(df_4.shape[0]):
  if df_4.iloc[i][5]=='operating':
    start_date = df_4.iloc[i][11]
    end_date = "2020-12-16"
    if type(start_date) == float or type(end_date) == float:
      continue
    start_date = str(start_date)
    st_dt = [int(it) for it in start_date.replace('-', ' ').split(' ')]
    ed_dt = [int(it) for it in end_date.replace('-', ' ').split(' ')]
    actual_age = actual_age.append({'permalink' : df_4.iloc[i][0], 'name' : df_4.iloc[i][1], 'age' : getDifference(Date(st_dt[2],st_
```

```
    actual_age = actual_age.append({'permalink': df_4.iloc[i][0], 'name': df_4.iloc[i][1], 'age': getDifference(Date(st_dt[2],st_
                ignore_index = True)
```

Data Cleaning

Concat all rows for NotAge Variables

data transformation needed hours of processing thus data was split and distributed among team members to process parallely

```
df1 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_1.csv')[0:4400]
df2 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_2.csv')[4400:8800]
df3 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_3.csv')[8800:13200]
df4 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_4.csv')[13200:17600]
df5 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_5.csv')[22000:26400]
df6 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_5 (1).csv')[17600:22000]
df7 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_6.csv')[26400:30800]
df8a = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_7a.csv')[30800:31900]
df8b = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_7b.csv')[31900:33000]
df8c = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_7c.csv')[33000:34100]
df8d = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_7d.csv')[34100:35200]
df9 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_8.csv')[35200:39600]
df10 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/NotAge21_66k_9.csv')[39600:44000]
df11 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/final - 1c.csv')[44000:]
```

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (19) have mixed types.Speci
  interactivity=interactivity, compiler=compiler, result=result)
```

```
df = pd.concat([df1,df2,df3,df4,df5,df6,df7,df8a,df8b,df8c,df8d,df9,df10,df11,])
df
```

t	roundA_raised_amount	total_investments	investment_per_round	funding_total_usd	total_acquisitions	competitors_count	compe
N	NaN	2.0	10000000.0	10000000.0	0.0	299.0	
N	NaN	4.0	350000.0	700000.0	0.0	19.0	
N	NaN	2.0	3406878.0	3406878.0	0.0	129.0	
N	2000000.0	1.0	2000000.0	2000000.0	0.0	1211.0	
N	NaN	1.0	NaN	NaN	0.0	4109.0	
..	...	...	...	...	...	...	
N	1587301.0	1.0	1587301.0	1587301.0	0.0	1135.0	
N	NaN	3.0	28576.0	114304.0	0.0	14.0	
N	NaN	2.0	NaN	NaN	0.0	NaN	
N	NaN	1.0	18192.0	18192.0	0.0	20.0	
N	NaN	1.0	14851.0	14851.0	0.0	62.0	

Add Age Variable Columns (left join by company name & permalink)

```
df_age1 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/actual_age.csv')
df_age2 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/success_age.csv')
df_age3 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/roundA_age.csv')
df_age4 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/roundB_age.csv')
df_age5 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/roundC_age.csv')
df_age6 = pd.read_csv('/content/drive/MyDrive/StartUp_Project/roundD_age.csv')
```

```
print(df_age2['permalink'].unique().shape, df_age1['permalink'].shape)
```

```
(1435,) (41414,)
```

```
df_age1.value_counts(subset=['name'])
```

```
name
吃神马 ChiShenMa               1
Glownet                       1
Globial                       1
Globehook                     1
Globeecom International       1
                             ..
Rachel Joyce Organic Salon    1
RacerTimes                    1
Racemi                        1
Race Yourself                 1
#HASHOFF                      1
Length: 41414, dtype: int64
```

```
df_age1.loc[df_age1['name']=='500px']
```

	permalink	name	actual_age
264	/organization/500px	500px	4094

```
df = df.merge(df_age1, how='left', on = ['permalink','name'])
df = df.merge(df_age2, how='left', on = ['permalink','name'])
df = df.merge(df_age3, how='left', on = ['permalink','name'])
df = df.merge(df_age4, how='left', on = ['permalink','name'])
df = df.merge(df_age5, how='left', on = ['permalink','name'])
df = df.merge(df_age6, how='left', on = ['permalink','name'])
```

Cleaning NaN Values

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

```
permalink                  0
name                       1
roundD                     0
roundC                     0
roundB                     0
roundA                     0
VentureCapital             0
IsTech                  3148
target                     0
roundD_raised_amount   64772
roundC_raised_amount   62798
roundB_raised_amount   59626
roundA_raised_amount   56112
total_investments          0
investment_per_round   12785
funding_total_usd      12785
total_acquisitions         0
competitors_count       3148
competitors_acquired    3148
country_code            6958
funding_rounds             0
investors_per_round        0
top500_investors       21630
actual_age             24954
success_age            61971
roundA_age             57273
roundD_age             64933
dtype: int64
```

```
#drop companies for which we dont have investments data
in_investments = list(investments['company_permalink'].unique())
for i in range(df.shape[0]):
  if df.loc[i,'permalink'] in in_investments:
    if df.loc[i, 'roundD'] == 0:
      df.loc[i,'roundD_raised_amount'] = 0
    if df.loc[i, 'roundC'] == 0:
      df.loc[i,'roundC_raised_amount'] = 0
    if df.loc[i, 'roundB'] == 0:
```

```
      df.loc[i,'roundB_raised_amount'] = 0
    if df.loc[i, 'roundA'] == 0:
      df.loc[i,'roundA_raised_amount'] = 0
  else:
    df = df.drop(index=i)
  print(i)
```

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

```
permalink                  0
name                       1
roundD                     0
roundC                     0
roundB                     0
roundA                     0
VentureCapital             0
IsTech                  2429
target                     0
roundD_raised_amount      90
roundC_raised_amount     187
roundB_raised_amount     469
roundA_raised_amount    1495
total_investments          0
investment_per_round    8537
funding_total_usd       8537
total_acquisitions         0
competitors_count       2429
competitors_acquired    2429
country_code            4990
funding_rounds             0
investors_per_round        0
top500_investors           0
actual_age             18637
success_age            40774
roundA_age             35643
roundD_age             43303
dtype: int64
```

```
# take inverse of ages as new variables so that we can replace nan values with zeros and scale up to preserve significant digits
df['success_age_inverse'] = 10000/df['success_age']
df['actual_age_inverse'] = 10000/df['actual_age']
df['roundD_age_inverse'] = 10000/df['roundD_age']
df['roundC_age_inverse'] = 10000/df['roundC_age']
df['roundB_age_inverse'] = 10000/df['roundB_age']
df['roundA_age_inverse'] = 10000/df['roundA_age']

#fill nan values with 0
df[['roundA_age_inverse','roundB_age_inverse','roundC_age_inverse','roundD_age_inverse','success_age_inverse']] = df[['roundA_age_inv
#drop original age columns
df = df.drop(columns = ['roundA_age','roundB_age','roundC_age','roundD_age','success_age','actual_age'])
```

```
## drop companies with no name
df = df.dropna(subset = ['name'])
```

```
# redefine IsTech to remove nan values
tech_keys = ['tech','analytics', 'software', 'elec', 'web','manufacturing','internet','auto','smart','e-','data','develop','product',
for i in df.index:
  for key in tech_keys:
    if key in df.loc[i,'name']:
      df.loc[i,'IsTech'] = 1
    else:
      df.loc[i,'IsTech'] = 0
    if (isinstance(companies.loc[i,'category_list'],str)):
      if key in companies.loc[i,'category_list']:
        df.loc[i,'IsTech'] = 1
      else:
        df.loc[i,'IsTech'] = 0
  print(i)
```

```
df
```

try_code	funding_rounds	investors_per_round	top500_investors	IsTech	succes_age_inverse	actual_age_inverse	roundD_age_inve
CHN	1.0	1.000000	0.0	0.0	NaN	0.000196	
USA	1.0	1.000000	0.0	0.0	NaN	0.000250	
HKG	1.0	1.000000	1.0	0.0	NaN	NaN	
USA	4.0	3.500000	4.0	0.0	NaN	0.000275	
USA	3.0	4.333333	7.0	0.0	NaN	0.000292	
...	...	...	...	...	...	...	
CHN	1.0	1.000000	1.0	0.0	NaN	NaN	
HRV	4.0	0.750000	3.0	0.0	NaN	0.000319	
NaN	1.0	2.000000	0.0	0.0	NaN	0.000275	
USA	1.0	1.000000	1.0	0.0	NaN	0.000394	
NaN	1.0	1.000000	0.0	0.0	NaN	NaN	

```
#adding 2 extra variables

## status = current status of company (operational not operational)
## last_funding_at = year of last funding
for i in df.index:
  df.loc[i,'last_funding_at'] = int(companies.loc[i,'last_funding_at'][0:4])
  df.loc[i,'status'] = companies.loc[i,'status']
```

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

```
permalink                 0
name                      0
roundD                    0
roundC                    0
roundB                    0
roundA                    0
VentureCapital            0
target                    0
roundD_raised_amount     90
roundC_raised_amount    187
roundB_raised_amount    469
roundA_raised_amount   1495
total_investments         0
investment_per_round   8537
funding_total_usd      8537
total_acquisitions        0
competitors_count      2429
competitors_acquired   2429
country_code           4990
funding_rounds            0
investors_per_round       0
top500_investors          0
IsTech                    0
actual_age_inverse    18636
roundD_age_inverse        0
roundC_age_inverse        0
roundB_age_inverse        0
roundA_age_inverse        0
success_age_inverse       0
last_funding_at           0
status                    0
dtype: int64
```

```
## competitor = 0 if not found
df[['competitors_count','competitors_acquired']] = df[['competitors_count','competitors_acquired']].fillna(0)
```

```
## take roundUp integer for investors_per_round
df['investors_per_round'] = df['investors_per_round'].apply(np.ceil)
```

```
df.to_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_raw.csv')
df
```

de	funding_rounds	investors_per_round	top500_investors	IsTech	actual_age_inverse	roundD_age_inverse	roundC_age_inverse	r
HN	1.0	1.0	0.0	0.0	1.961554	0.0	0.0	
SA	1.0	1.0	0.0	0.0	2.498751	0.0	0.0	
KG	1.0	1.0	1.0	0.0	NaN	0.0	0.0	
SA	4.0	4.0	4.0	0.0	2.749519	0.0	0.0	
SA	3.0	5.0	7.0	0.0	2.919708	0.0	0.0	
...	...	...	...	...	...	...	...	
HN	1.0	1.0	1.0	0.0	NaN	0.0	0.0	
RV	4.0	1.0	3.0	0.0	3.185728	0.0	0.0	
aN	1.0	2.0	0.0	0.0	2.749519	0.0	0.0	
SA	1.0	1.0	1.0	0.0	3.935458	0.0	0.0	
aN	1.0	1.0	0.0	0.0	NaN	0.0	0.0	

```
df['actual_age_inverse'] = df['actual_age_inverse'].fillna(df['actual_age_inverse'].median())
df_clean = df.dropna()
df_clean
```

	permalink	name	roundD	roundC	roundB	roundA	VentureCapital	target	roundD_raised_amount	roundC_ra
0	/organization/0-6-com	0-6.com	0.0	0.0	0.0	1.0	1.0	0.0	0.0	
2	/organization/01games-technology	01Games Technology	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	/organization/0xdata	H2O.ai	0.0	0.0	1.0	1.0	1.0	0.0	0.0	
4	/organization/1	One Inc.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5	/organization/1-2-3-listo	1,2,3 Listo	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
44730	/organization/zytoprotec	Zytoprotec	0.0	0.0	0.0	1.0	1.0	0.0	0.0	
44731	/organization/zzish	Zzish	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
44732	/organization/zznode-science-and-technology-co...	ZZNode Science and Technology	0.0	0.0	0.0	1.0	1.0	0.0	0.0	
44733	/organization/zzzzapp-com	Zzzzapp Wireless ltd.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
44735	/organization/Ôasys-2	Ôasys	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

32306 rows × 31 columns

```
df_clean.isnull().sum()
```

```
permalink            0
name                 0
roundD               0
roundC               0
roundB               0
roundA               0
VentureCapital       0
```

```
target                   0
roundD_raised_amount     0
roundC_raised_amount     0
roundB_raised_amount     0
roundA_raised_amount     0
total_investments        0
investment_per_round     0
funding_total_usd        0
total_acquisitions       0
competitors_count        0
competitors_acquired     0
country_code             0
funding_rounds           0
investors_per_round      0
top500_investors         0
IsTech                   0
actual_age_inverse       0
roundD_age_inverse       0
roundC_age_inverse       0
roundB_age_inverse       0
roundA_age_inverse       0
success_age_inverse      0
last_funding_at          0
status                   0
dtype: int64
```

```
df_clean.to_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_cleaned.csv')
```

Data Visualization

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

```
df= pd.read_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_cleaned.csv', index_col=0)
#df2= pd.read_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_raw.csv', index_col=0)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32306 entries, 0 to 44735
Data columns (total 31 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   permalink             32306 non-null  object
 1   name                  32306 non-null  object
 2   roundD                32306 non-null  float64
 3   roundC                32306 non-null  float64
 4   roundB                32306 non-null  float64
 5   roundA                32306 non-null  float64
 6   VentureCapital        32306 non-null  float64
 7   target                32306 non-null  float64
 8   roundD_raised_amount  32306 non-null  float64
 9   roundC_raised_amount  32306 non-null  float64
 10  roundB_raised_amount  32306 non-null  float64
 11  roundA_raised_amount  32306 non-null  float64
 12  total_investments     32306 non-null  float64
 13  investment_per_round  32306 non-null  float64
 14  funding_total_usd     32306 non-null  float64
 15  total_acquisitions    32306 non-null  float64
 16  competitors_count     32306 non-null  float64
 17  competitors_acquired  32306 non-null  float64
 18  country_code          32306 non-null  object
 19  funding_rounds        32306 non-null  float64
 20  investors_per_round   32306 non-null  float64
 21  top500_investors      32306 non-null  float64
 22  IsTech                32306 non-null  float64
 23  actual_age_inverse    32306 non-null  float64
 24  roundD_age_inverse    32306 non-null  float64
 25  roundC_age_inverse    32306 non-null  float64
 26  roundB_age_inverse    32306 non-null  float64
 27  roundA_age_inverse    32306 non-null  float64
 28  success_age_inverse   32306 non-null  float64
 29  last_funding_at       32306 non-null  float64
 30  status                32306 non-null  object
```

```
dtypes: float64(27), object(4)
memory usage: 7.9+ MB
```

```
#numerical columns
df_num= df.drop(['permalink', 'name', 'country_code', 'status'], axis = 1)
df_num = df_num.astype('int64')
df_num
```

roundD	roundC	roundB	roundA	VentureCapital	target	roundD_raised_amount	roundC_raised_amount	roundB_raised_amount	round
0	0	0	1	1	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	1	1	1	0	0	0	20000000	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
0	0	0	1	1	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	1	1	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	

ows × 27 columns

```
list(df_num.columns)
```

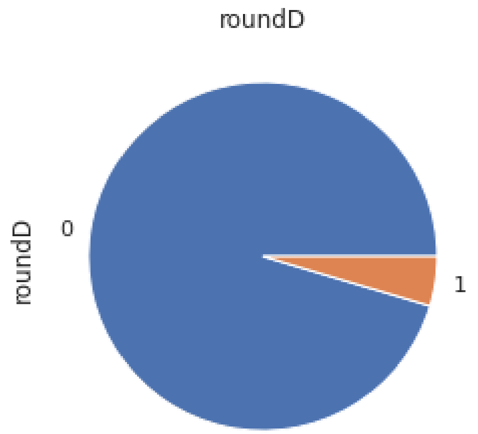
```
['roundD',
 'roundC',
 'roundB',
 'roundA',
```

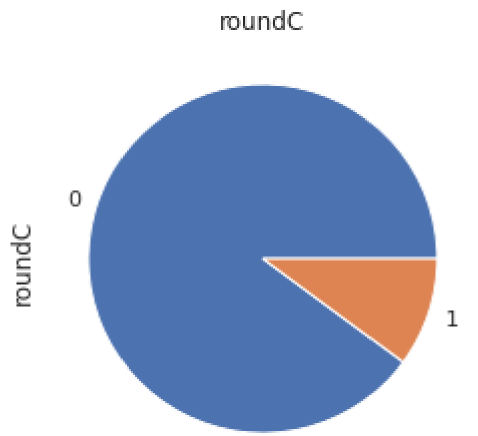
```
     'VentureCapital',
     'target',
     'roundD_raised_amount',
     'roundC_raised_amount',
     'roundB_raised_amount',
     'roundA_raised_amount',
     'total_investments',
     'investment_per_round',
     'funding_total_usd',
     'total_acquisitions',
     'competitors_count',
     'competitors_acquired',
     'funding_rounds',
     'investors_per_round',
     'top500_investors',
     'IsTech',
     'actual_age_inverse',
     'roundD_age_inverse',
     'roundC_age_inverse',
     'roundB_age_inverse',
     'roundA_age_inverse',
     'success_age_inverse',
     'last_funding_at']
```

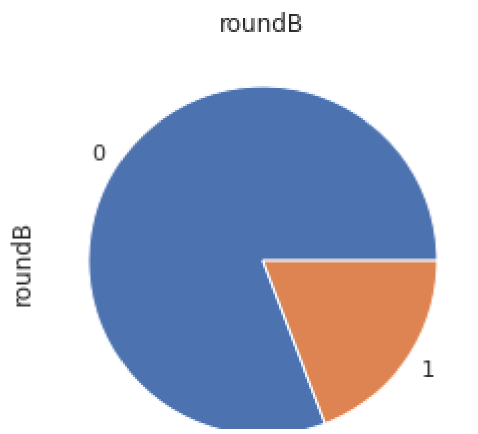
```
df['status'].value_counts()[:].plot(kind='pie')
```

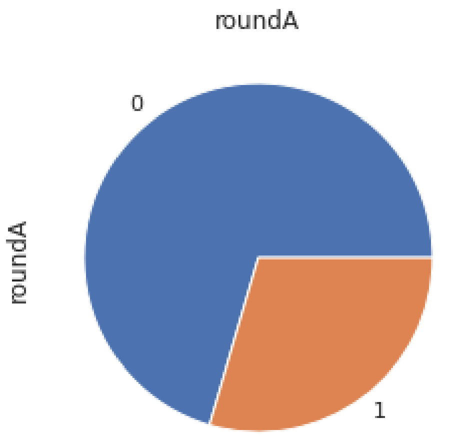
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f12fc17eba8>
```

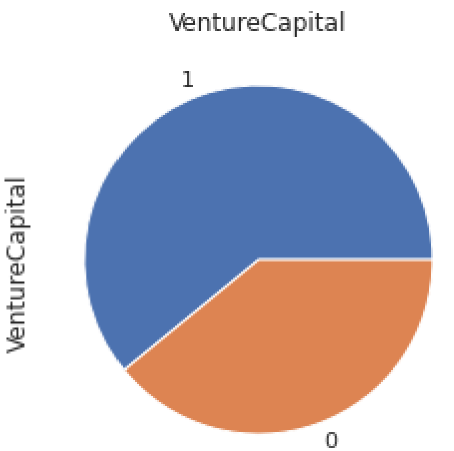
```
for i in df_num.columns:
  if (df_num[i].unique().shape[0]) <= 5:
    df_num[i].value_counts()[:].plot(kind='pie')
    plt.title(i)
    plt.show()
  else:
    plt.hist(df_num[i])
    plt.title(i)
    plt.show()
```

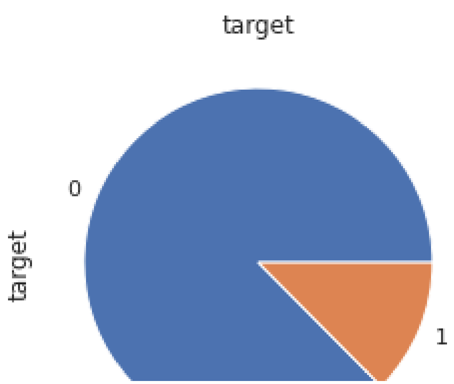

roundD
roundD
0
1

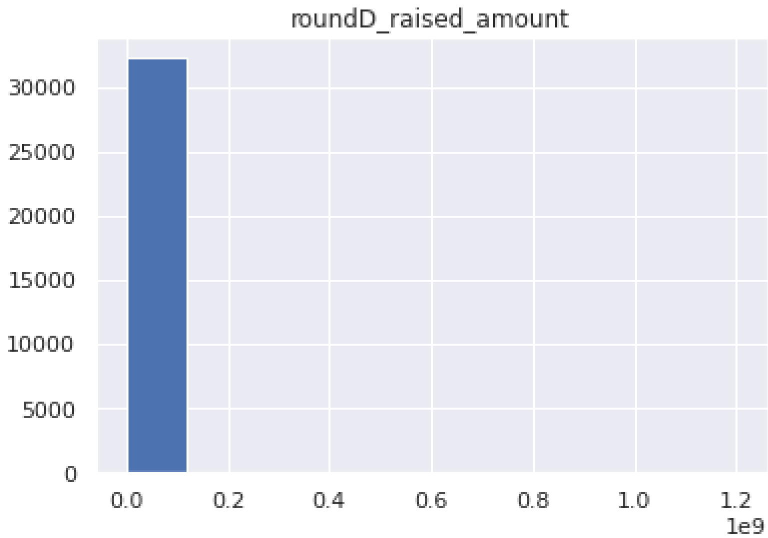

roundC
roundC
0
1

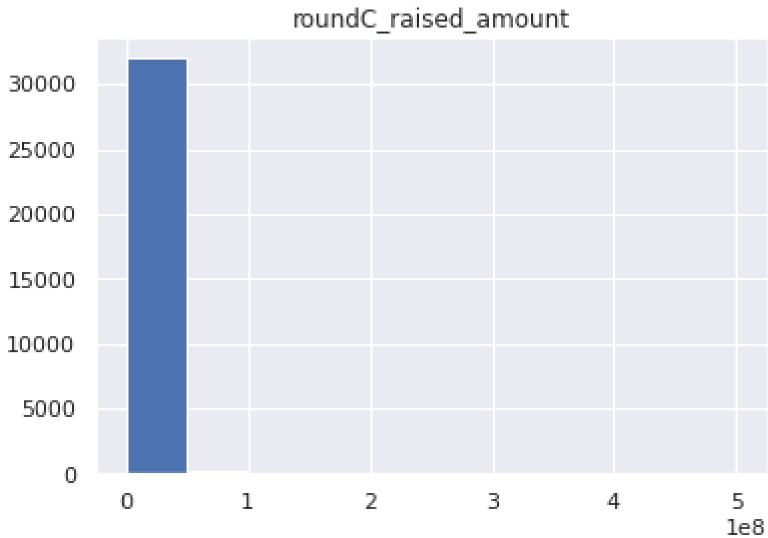

roundB
roundB
0
1

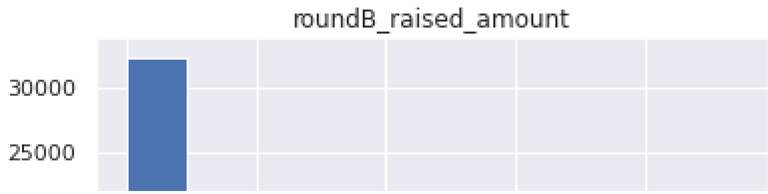

roundA
0
roundA
1

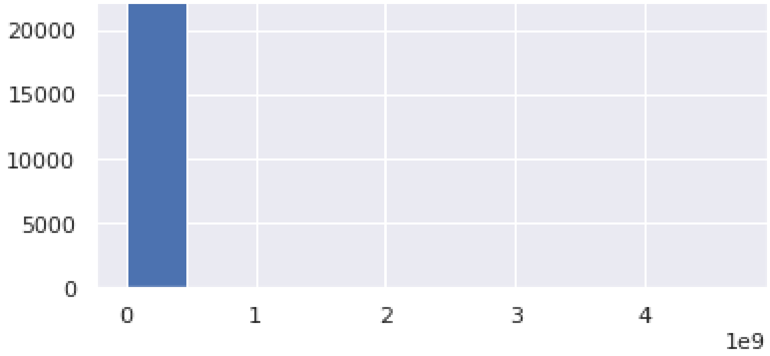

VentureCapital
1
VentureCapital
0

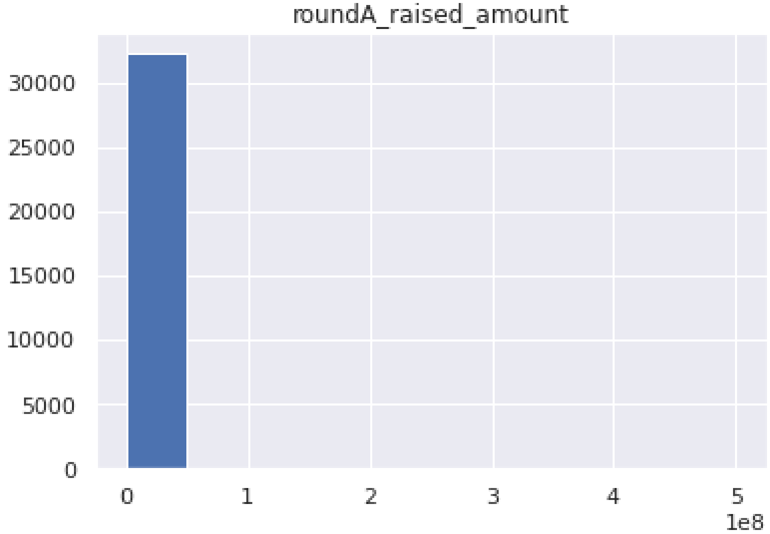

target
0
target
1

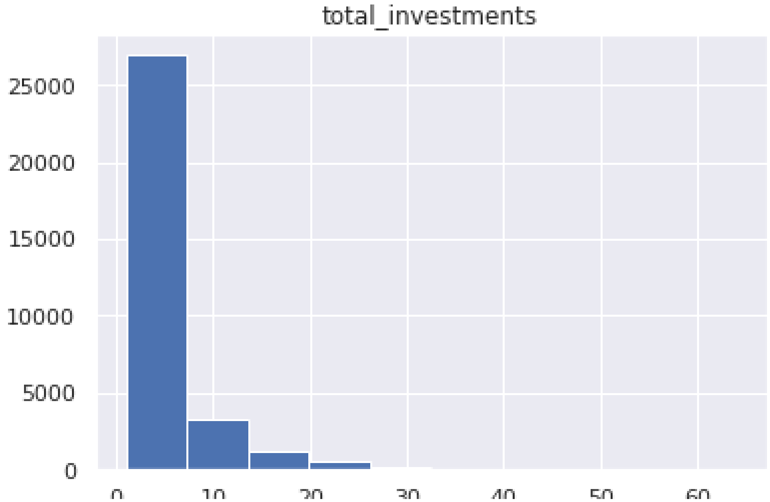

roundD_raised_amount
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e9

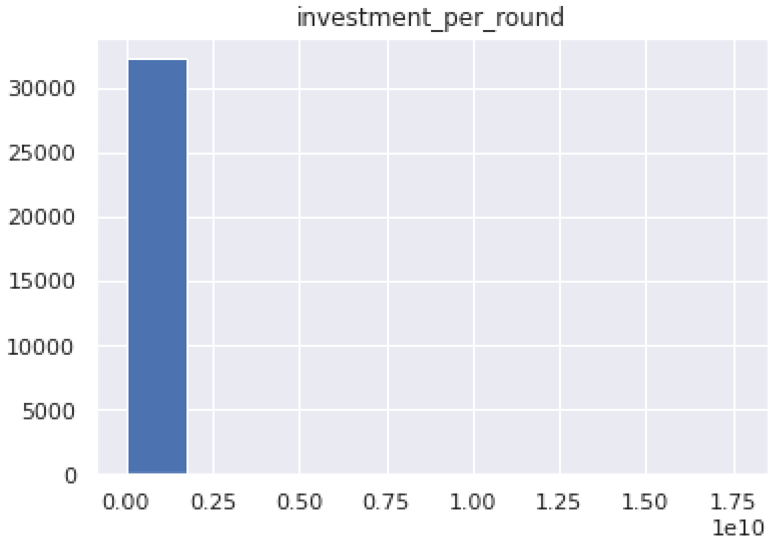

roundC_raised_amount
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
1e8

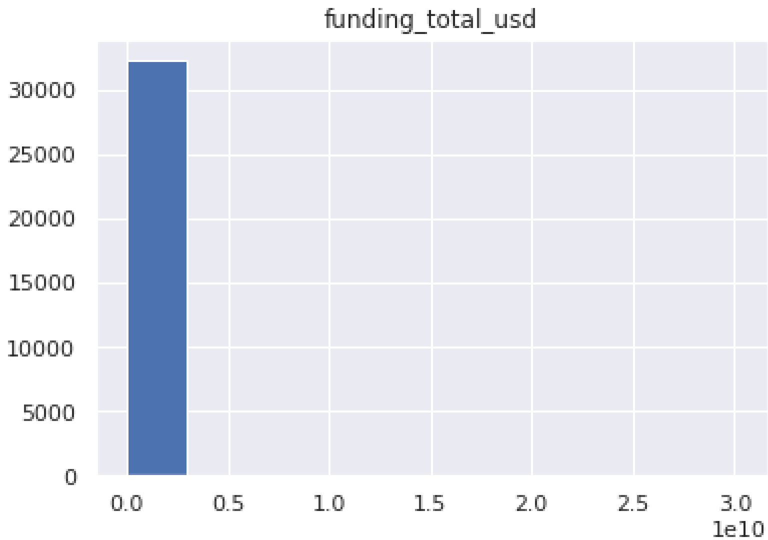

roundB_raised_amount
30000
25000

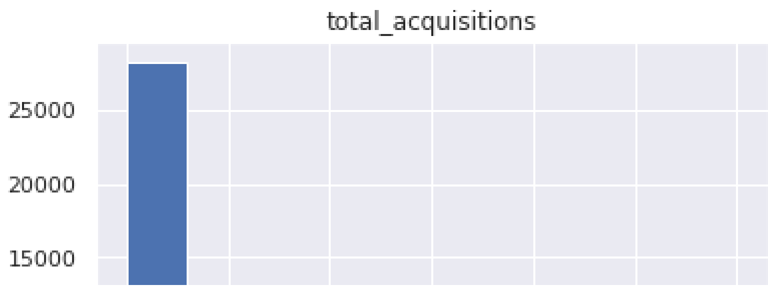

20000
15000
10000
5000
0
0
1
2
3
4
1e9

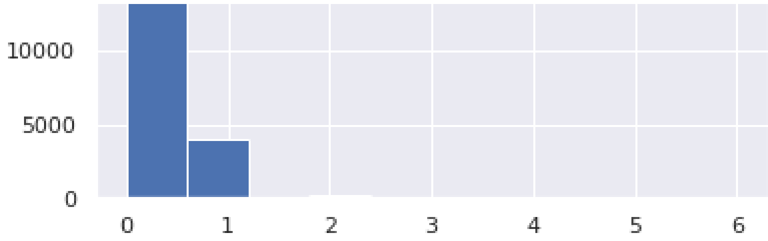

roundA_raised_amount
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
1e8


total_investments
25000
20000
15000
10000
5000
0

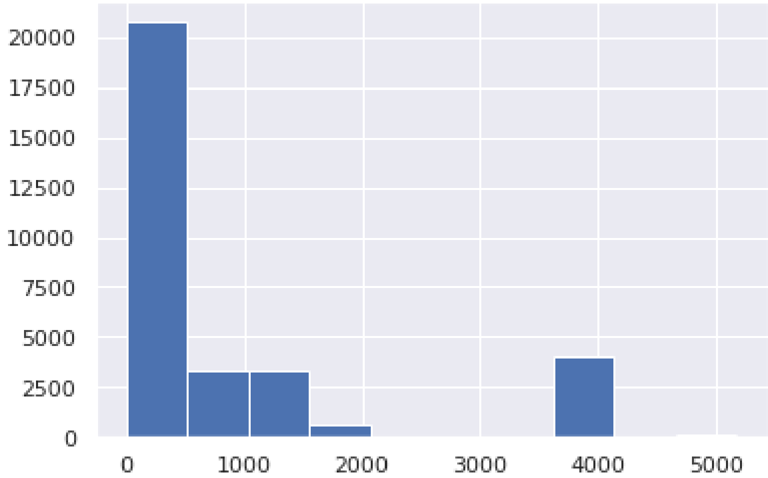

investment_per_round
30000
25000
20000
15000
10000
5000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e10


funding_total_usd
30000
25000
20000
15000
10000
5000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e10

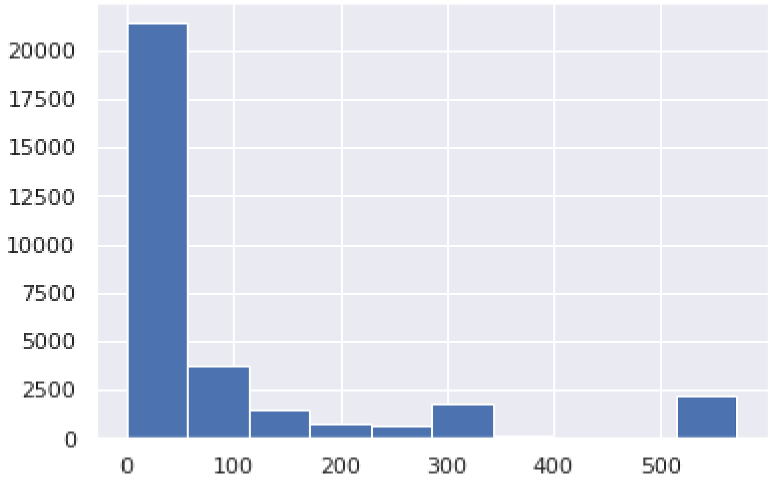

total_acquisitions
25000
20000
15000



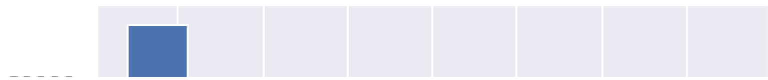
competitors_count

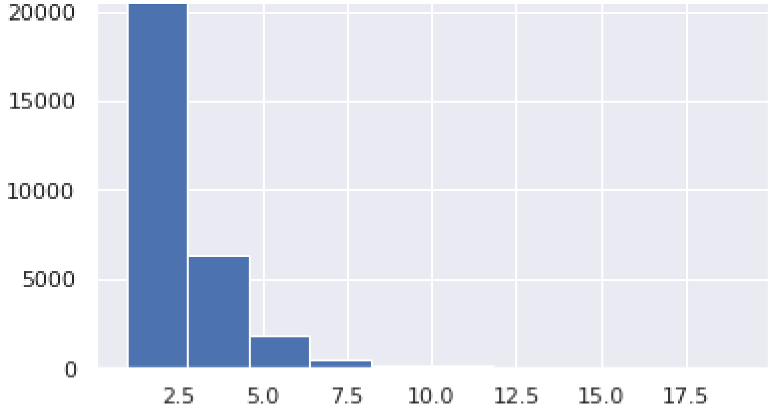


competitors_acquired

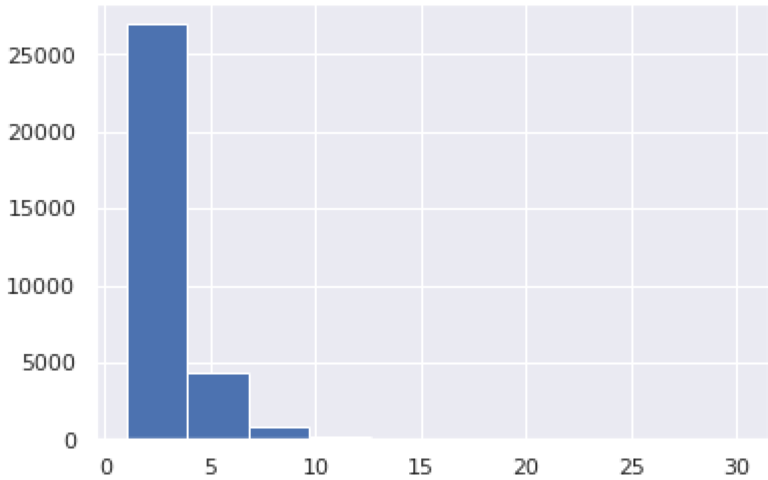


funding_rounds

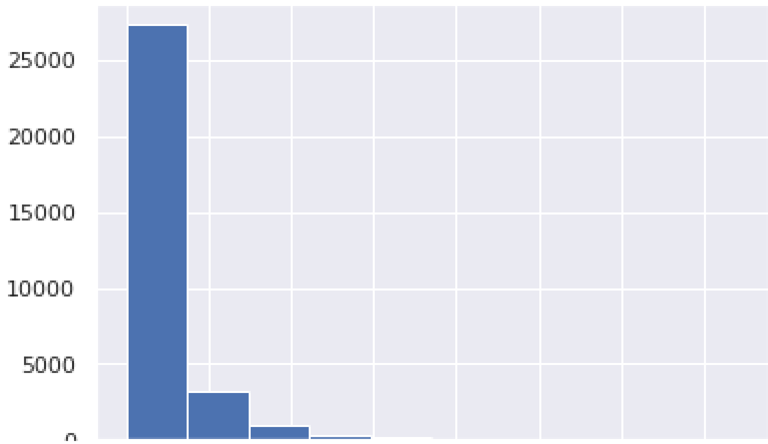




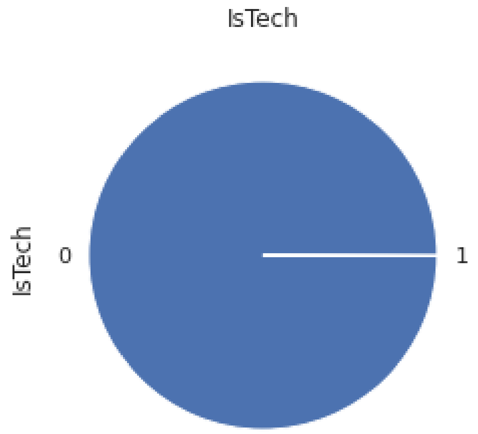
investors_per_round

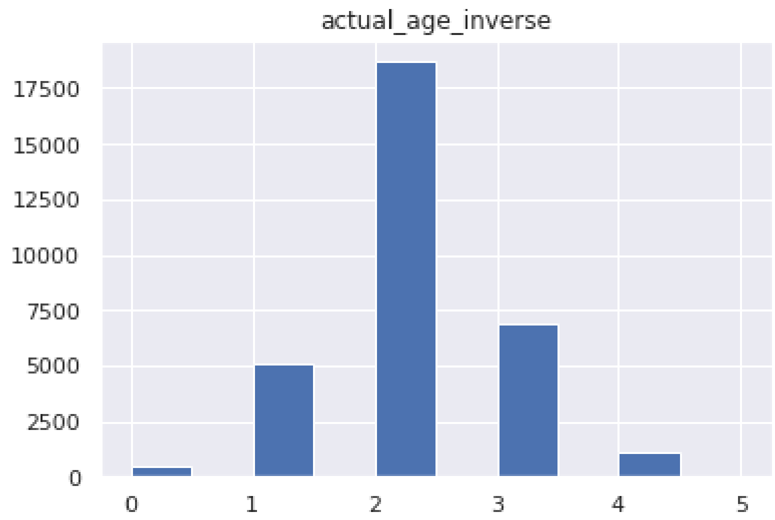


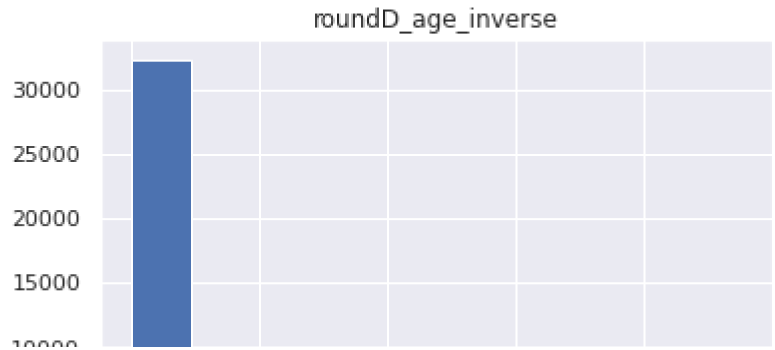
top500_investors

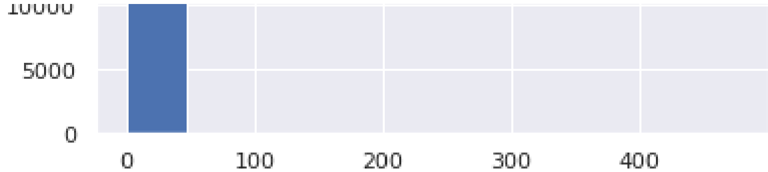



0
0
5
10
15
20
25
30
35

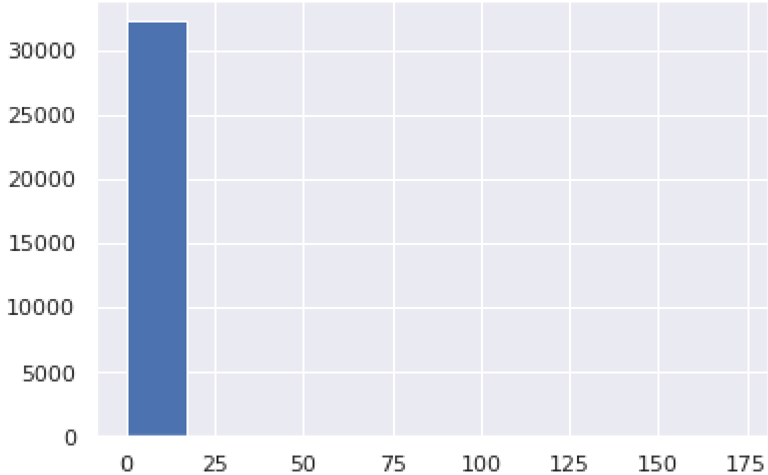

IsTech
IsTech
0
1


actual_age_inverse
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5

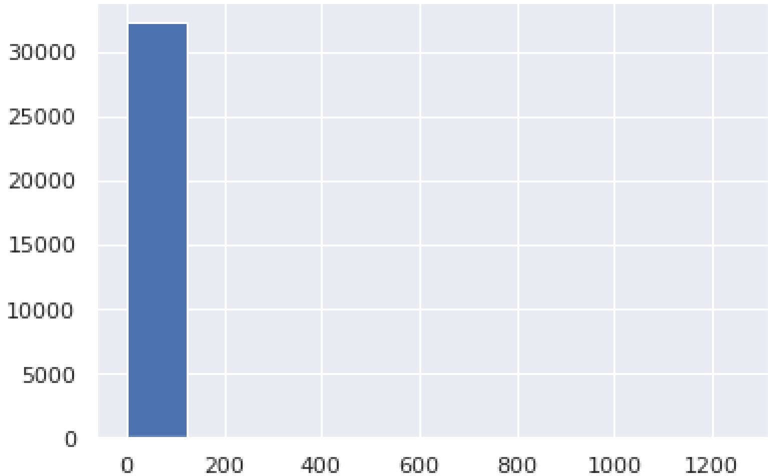

roundD_age_inverse
30000
25000
20000
15000



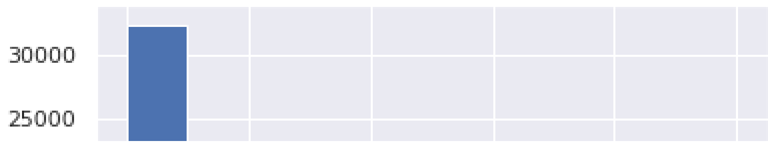
roundC_age_inverse

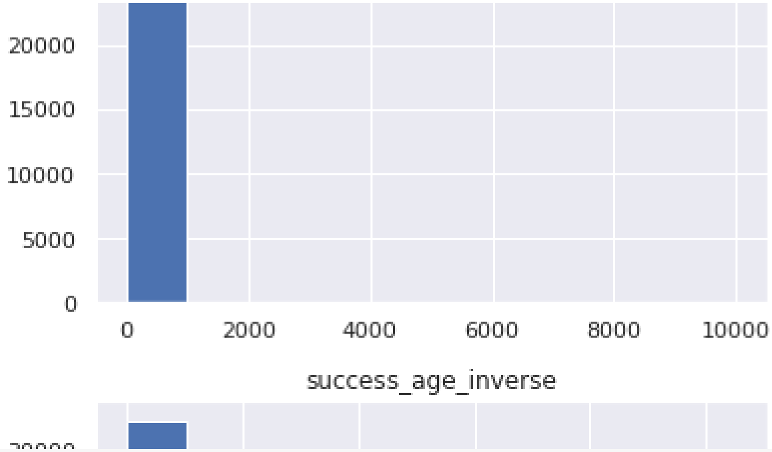


roundB_age_inverse



roundA_age_inverse



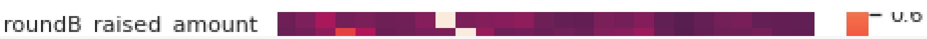



```
#correlation plot
print(df_num.corr())
sns.heatmap(df_num.corr())
```

```
                         roundD    roundC  ...  success_age_inverse  last_funding_at
roundD                 1.000000  0.401758  ...            -0.000102         0.002951
roundC                 0.401758  1.000000  ...             0.014395         0.001184
roundB                 0.200342  0.359153  ...             0.018651        -0.000024
roundA                 0.047889  0.105287  ...             0.036630        -0.008181
VentureCapital         0.175929  0.268616  ...             0.015236        -0.004684
target                 0.088053  0.135882  ...             0.370301        -0.000154
roundD_raised_amount   0.453083  0.214046  ...            -0.001222        -0.001342
roundC_raised_amount   0.237302  0.609607  ...             0.000108        -0.000293
roundB_raised_amount   0.034525  0.089407  ...             0.005318         0.001154
roundA_raised_amount   0.018394  0.049086  ...             0.006486        -0.007825
total_investments      0.389214  0.425100  ...             0.035646        -0.005224
investment_per_round   0.022259  0.024528  ...            -0.003190         0.000746
funding_total_usd      0.093968  0.089066  ...            -0.003327         0.004104
total_acquisitions     0.085524  0.132119  ...             0.362379        -0.000004
competitors_count      0.041189  0.057136  ...            -0.012390        -0.001806
competitors_acquired   0.038448  0.057766  ...             0.001424         0.000814
funding_rounds         0.356270  0.395135  ...             0.003291         0.005408
investors_per_round    0.144991  0.183733  ...             0.058426        -0.007335
top500_investors       0.416561  0.451100  ...             0.046574        -0.007096
IsTech                -0.006113 -0.001960  ...            -0.002740        -0.001620
actual_age_inverse    -0.129153 -0.166923  ...            -0.018393         0.003915
roundD_age_inverse     0.304668  0.140277  ...             0.001908         0.005123
roundC_age_inverse     0.316830  0.632523  ...             0.023781        -0.000175
roundB_age_inverse     0.102984  0.155651  ...             0.023006         0.001786
roundA_age_inverse     0.006933  0.026742  ...             0.030446         0.005111
success_age_inverse   -0.000102  0.014395  ...             1.000000         0.003832
last_funding_at        0.002951  0.001184  ...             0.003832         1.000000

[27 rows x 27 columns]
<matplotlib.axes._subplots.AxesSubplot at 0x7f12f3294748>
```

XG-Boost Classification




```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot
import joblib

#XGB handles missing values very well, thus using raw data itself
df = pd.read_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_raw.csv', index_col=0)

X = df.drop(columns = ['success_age_inverse','total_acquisitions','target', 'name', 'permalink'])
Y = df.target
```

```
le = LabelEncoder()

#transform country_code
X['country_code']=X['country_code'].map(str)
le.fit(X['country_code'])
X['country_code'] = le.transform(X['country_code'])
joblib.dump(le, '/content/drive/MyDrive/StartUp_Project/country_code_label.joblib')

#transfor status
le.fit(X['status'])
X['status'] = le.transform(X['status'])
joblib.dump(le, '/content/drive/MyDrive/StartUp_Project/status_label.joblib')
```

```
['/content/drive/MyDrive/StartUp_Project/status_label.joblib']
```

```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=7)
# fit model on training data
model = XGBClassifier(max_depth=70,n_estimators=150,learning_rate= 0.05)

eval_set = [(X_train, y_train), (X_test, y_test)]

model.fit(X_train, y_train,  early_stopping_rounds=15,eval_metric=["error", "logloss"], eval_set=eval_set, verbose=True)
```

```
[55]	validation_0-error:0.010869	validation_0-logloss:0.087473	validation_1-error:0.070295	validation_1-logloss
[56]	validation_0-error:0.010702	validation_0-logloss:0.085446	validation_1-error:0.070407	validation_1-logloss
[57]	validation_0-error:0.010562	validation_0-logloss:0.08354	validation_1-error:0.070295	validation_1-logloss
[58]	validation_0-error:0.010282	validation_0-logloss:0.081656	validation_1-error:0.070072	validation_1-logloss
[59]	validation_0-error:0.010143	validation_0-logloss:0.079882	validation_1-error:0.070407	validation_1-logloss
```

```
[60]	validation_0-error:0.009919	validation_0-logloss:0.078183	validation_1-error:0.069736	validation_1-logloss
[61]	validation_0-error:0.009752	validation_0-logloss:0.076547	validation_1-error:0.06996	validation_1-logloss
[62]	validation_0-error:0.009696	validation_0-logloss:0.074863	validation_1-error:0.069848	validation_1-logloss
[63]	validation_0-error:0.009416	validation_0-logloss:0.073141	validation_1-error:0.069848	validation_1-logloss
[64]	validation_0-error:0.009221	validation_0-logloss:0.071535	validation_1-error:0.069401	validation_1-logloss
[65]	validation_0-error:0.008969	validation_0-logloss:0.070041	validation_1-error:0.069177	validation_1-logloss
[66]	validation_0-error:0.008718	validation_0-logloss:0.068539	validation_1-error:0.069289	validation_1-logloss
[67]	validation_0-error:0.008578	validation_0-logloss:0.067167	validation_1-error:0.068507	validation_1-logloss
[68]	validation_0-error:0.00841	validation_0-logloss:0.065802	validation_1-error:0.068507	validation_1-logloss
[69]	validation_0-error:0.008271	validation_0-logloss:0.064422	validation_1-error:0.067836	validation_1-logloss
[70]	validation_0-error:0.008047	validation_0-logloss:0.063028	validation_1-error:0.067389	validation_1-logloss
[71]	validation_0-error:0.007963	validation_0-logloss:0.061653	validation_1-error:0.067166	validation_1-logloss
[72]	validation_0-error:0.007796	validation_0-logloss:0.060384	validation_1-error:0.067389	validation_1-logloss
[73]	validation_0-error:0.007656	validation_0-logloss:0.059231	validation_1-error:0.067166	validation_1-logloss
[74]	validation_0-error:0.007488	validation_0-logloss:0.058113	validation_1-error:0.066942	validation_1-logloss
[75]	validation_0-error:0.007405	validation_0-logloss:0.056955	validation_1-error:0.066272	validation_1-logloss
[76]	validation_0-error:0.007293	validation_0-logloss:0.055897	validation_1-error:0.066384	validation_1-logloss
[77]	validation_0-error:0.007181	validation_0-logloss:0.054873	validation_1-error:0.06616	validation_1-logloss
[78]	validation_0-error:0.007125	validation_0-logloss:0.053849	validation_1-error:0.065937	validation_1-logloss
[79]	validation_0-error:0.006874	validation_0-logloss:0.052862	validation_1-error:0.065378	validation_1-logloss
[80]	validation_0-error:0.00665	validation_0-logloss:0.052063	validation_1-error:0.065601	validation_1-logloss
[81]	validation_0-error:0.006371	validation_0-logloss:0.051136	validation_1-error:0.065378	validation_1-logloss
[82]	validation_0-error:0.006343	validation_0-logloss:0.050259	validation_1-error:0.064931	validation_1-logloss
[83]	validation_0-error:0.006231	validation_0-logloss:0.049543	validation_1-error:0.065266	validation_1-logloss
[84]	validation_0-error:0.006203	validation_0-logloss:0.048687	validation_1-error:0.064484	validation_1-logloss
[85]	validation_0-error:0.006035	validation_0-logloss:0.048002	validation_1-error:0.064372	validation_1-logloss
[86]	validation_0-error:0.005896	validation_0-logloss:0.047376	validation_1-error:0.064148	validation_1-logloss
[87]	validation_0-error:0.005812	validation_0-logloss:0.046589	validation_1-error:0.063701	validation_1-logloss
[88]	validation_0-error:0.005784	validation_0-logloss:0.045939	validation_1-error:0.063813	validation_1-logloss
[89]	validation_0-error:0.005616	validation_0-logloss:0.045336	validation_1-error:0.063813	validation_1-logloss
[90]	validation_0-error:0.005588	validation_0-logloss:0.044648	validation_1-error:0.064037	validation_1-logloss
[91]	validation_0-error:0.005477	validation_0-logloss:0.043962	validation_1-error:0.063478	validation_1-logloss
[92]	validation_0-error:0.005337	validation_0-logloss:0.043398	validation_1-error:0.06359	validation_1-logloss
[93]	validation_0-error:0.005309	validation_0-logloss:0.042844	validation_1-error:0.063254	validation_1-logloss
[94]	validation_0-error:0.005281	validation_0-logloss:0.042328	validation_1-error:0.063031	validation_1-logloss
[95]	validation_0-error:0.005169	validation_0-logloss:0.041687	validation_1-error:0.062919	validation_1-logloss
[96]	validation_0-error:0.005085	validation_0-logloss:0.041176	validation_1-error:0.063478	validation_1-logloss
[97]	validation_0-error:0.004918	validation_0-logloss:0.040559	validation_1-error:0.062807	validation_1-logloss
[98]	validation_0-error:0.004834	validation_0-logloss:0.040026	validation_1-error:0.062249	validation_1-logloss
[99]	validation_0-error:0.004778	validation_0-logloss:0.039602	validation_1-error:0.062137	validation_1-logloss
[100]	validation_0-error:0.004722	validation_0-logloss:0.039077	validation_1-error:0.061913	validation_1-logloss
[101]	validation_0-error:0.004638	validation_0-logloss:0.038691	validation_1-error:0.062137	validation_1-logloss
```

```
[102]   validation_0-error:0.004638    validation_0-logloss:0.038306   validation_1-error:0.062025    validation_1-logloss
[103]   validation_0-error:0.004638    validation_0-logloss:0.037802   validation_1-error:0.062025    validation_1-logloss
[104]   validation_0-error:0.004527    validation_0-logloss:0.037348   validation_1-error:0.062025    validation_1-logloss
[105]   validation_0-error:0.004471    validation_0-logloss:0.036974   validation_1-error:0.062137    validation_1-logloss
[106]   validation_0-error:0.004387    validation_0-logloss:0.036613   validation_1-error:0.062025    validation_1-logloss
[107]   validation_0-error:0.004415    validation_0-logloss:0.036167   validation_1-error:0.062025    validation_1-logloss
[108]   validation_0-error:0.004387    validation_0-logloss:0.035742   validation_1-error:0.062137    validation_1-logloss
[109]   validation_0-error:0.004359    validation_0-logloss:0.035335   validation_1-error:0.06169     validation_1-logloss
[110]   validation_0-error:0.004331    validation_0-logloss:0.035005   validation_1-error:0.061578    validation_1-logloss
[111]   validation_0-error:0.004275    validation_0-logloss:0.034654   validation_1-error:0.061354    validation_1-logloss
[112]   validation_0-error:0.004219    validation_0-logloss:0.034281   validation_1-error:0.060684    validation_1-logloss
[113]   validation_0-error:0.004163    validation_0-logloss:0.033964   validation_1-error:0.060907    validation_1-logloss
```

```
# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Test Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Test Accuracy: 93.94%
```

```
# retrieve performance metrics
results = model.evals_result()
epochs = len(results['validation_0']['error'])
x_axis = range(0, epochs)

# plot log loss
fig, ax = pyplot.subplots()
ax.plot(x_axis, results['validation_0']['logloss'], label='Train')
ax.plot(x_axis, results['validation_1']['logloss'], label='Test')
ax.legend()
pyplot.ylabel('Log Loss')
pyplot.xlabel('Epochs')
pyplot.title('XGBoost Log Loss')
pyplot.show()

# plot classification error
```

```
# plot classification error
fig, ax = pyplot.subplots()
ax.plot(x_axis, results['validation_0']['error'], label='Train')
ax.plot(x_axis, results['validation_1']['error'], label='Test')
ax.legend()
pyplot.ylabel('Classification Error')
pyplot.xlabel('Epochs')
pyplot.title('XGBoost Classification Error')
pyplot.show()
```


```
from tabulate import tabulate
cm = confusion_matrix(y_pred = predictions,y_true = y_test[:], labels=[0,1])
rows = np.array([["Actual Failures"], ["Actual Successfull"]])
rows = np.concatenate((rows,cm), axis=1)
headers = ["No. of Samples", "Predicted Failure", "Predicted Successfull"]
table = tabulate(rows,headers, tablefmt="grid")
print(table)
```

```
+--------------------+---------------------+-------------------------+
| No. of Samples     |   Predicted Failure |   Predicted Successfull |
+====================+=====================+=========================+
| Actual Failures    |                7702 |                     177 |
+--------------------+---------------------+-------------------------+
| Actual Successfull |                 365 |                     704 |
+--------------------+---------------------+-------------------------+
```

Training Scores

```
y_pred = model.predict(X_train)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_train, predictions)
print("Train Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Train Accuracy: 99.60%
```

```
from tabulate import tabulate
cm = confusion_matrix(y_pred = predictions,y_true = y_train[:], labels=[0,1])
rows = np.array([["Actual Failures"], ["Actual Successfull"]])
rows = np.concatenate((rows,cm), axis=1)
headers = ["No. of Samples", "Predicted Failure", "Predicted Successfull"]
table = tabulate(rows,headers, tablefmt="grid")
print(table)
```

```
+--------------------+---------------------+-------------------------+
| No. of Samples     |   Predicted Failure |   Predicted Successfull |
+====================+=====================+=========================+
| Actual Failures    |               31701 |                      10 |
+--------------------+---------------------+-------------------------+
| Actual Successfull |                 134 |                    3944 |
+--------------------+---------------------+-------------------------+
```

```
model.save_model('/content/drive/MyDrive/StartUp_Project/xgb_model.json')
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

df = pd.read_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_cleaned.csv', index_col=0)
X = df.drop(columns = ['success_age_inverse','total_acquisitions','target', 'name', 'permalink'])
Y = df.target
```

```
le = joblib.load('/content/drive/MyDrive/StartUp_Project/country_code_label.joblib')
X['country_code'] = X['country_code'].map(str)
X['country_code'] = le.transform(X['country_code'])

le = joblib.load('/content/drive/MyDrive/StartUp_Project/status_label.joblib')
X['status'] = le.transform(X['status'])
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=13)

# define class weights
w = {0:1, 1:1}
# define model
lgr = LogisticRegression(random_state=13, class_weight=w)
# fit it
lgr.fit(X_train,y_train)
```

```
lgr.fit(X_train,y_train)
# test
y_pred = lgr.predict(X_test)
# performance
print(f'Test Accuracy Score: {accuracy_score(y_test,y_pred)}')
```

```
Test Accuracy Score: 0.8687712782420304
```

```
cm = confusion_matrix(y_pred = y_pred,y_true = y_test[:], labels=[0,1])
rows = np.array([["Actual Failures"], ["Actual Successfull"]])
rows = np.concatenate((rows,cm), axis=1)
headers = ["No. of Samples", "Predicted Failure", "Predicted Successfull"]
table = tabulate(rows,headers, tablefmt="grid")
print(table)
```

```
+--------------------+---------------------+-------------------------+
| No. of Samples     |   Predicted Failure |   Predicted Successfull |
+====================+=====================+=========================+
| Actual Failures    |                5613 |                       9 |
+--------------------+---------------------+-------------------------+
| Actual Successfull |                 839 |                       1 |
+--------------------+---------------------+-------------------------+
```

Training Scores

```
y_pred = lgr.predict(X_train)
# performance
print(f'Train Accuracy Score: {accuracy_score(y_train,y_pred)}')
```

```
Train Accuracy Score: 0.8750967342516638
```

```
cm = confusion_matrix(y_pred = y_pred,y_true = y_train[:], labels=[0,1])
rows = np.array([["Actual Failures"], ["Actual Successfull"]])
rows = np.concatenate((rows,cm), axis=1)
headers = ["No. of Samples", "Predicted Failure", "Predicted Successfull"]
table = tabulate(rows,headers, tablefmt="grid")
```

```
print(table)
```

```
+--------------------+---------------------+-------------------------+
| No. of Samples     |   Predicted Failure |   Predicted Successfull |
+====================+=====================+=========================+
| Actual Failures    |               22614 |                      17 |
+--------------------+---------------------+-------------------------+
| Actual Successfull |                3211 |                       2 |
+--------------------+---------------------+-------------------------+
```

```
joblib.dump(lgr, '/content/drive/MyDrive/StartUp_Project/lgr_model.joblib')
```

```
['/content/drive/MyDrive/StartUp_Project/lgr_model.joblib']
```

Data Imputation

To get good results with MLP we attempt to regain lost datapoint due to null values by imputing missing values.

Datawig is a package that does this job us. Based on deep learning techniques it imputes missing values of columns by using correlations & training NNs with other columns

https://github.com/awslabs/datawig

```
pip install datawig
```

```
import datawig
df_imputed = datawig.SimpleImputer.complete(df)
```

```
df_imputed.to_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_cleaned.csv')
```

MLP

```
# Binary Classification with Keras Neural Network
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold

# load dataset
df = pd.read_csv('/content/drive/MyDrive/StartUp_Project/Stratup_AllVariables_cleaned.csv', index_col=0)
dataset = df.values

X = df.drop(columns = ['success_age_inverse','total_acquisitions','target', 'name', 'permalink'])
Y = df.target

# encode class values as integers
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
```

```
le = joblib.load('/content/drive/MyDrive/StartUp_Project/country_code_label.joblib')
X['country_code'] = X['country_code'].map(str)
X['country_code'] = le.transform(X['country_code'])

le = joblib.load('/content/drive/MyDrive/StartUp_Project/status_label.joblib')
X['status'] = le.transform(X['status'])
```

```
X_train, X_test, y_train, y_test = train_test_split(X, encoded_Y, test_size=0.2, random_state=13)
```

BaseLine Model with 2 Layers

```
# baseline model
```

```
def create_baseline():
  # create model
  model = Sequential()
  model.add(Dense(32, input_dim=26, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  # Compile model
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
```

```
# evaluate model with dataset
estimator = KerasClassifier(build_fn=create_baseline, epochs=100, batch_size=5, verbose=2)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X_train, y_train, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
Epoch 23/100
5815/5815 - 5s - loss: 0.6062 - accuracy: 0.8745
Epoch 24/100
5815/5815 - 5s - loss: 0.3777 - accuracy: 0.8745
Epoch 25/100
5815/5815 - 5s - loss: 0.3777 - accuracy: 0.8745
Epoch 26/100
5815/5815 - 5s - loss: 0.3777 - accuracy: 0.8745
Epoch 27/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 28/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 29/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 30/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 31/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 32/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 33/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 34/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 35/100
```

```
Epoch 35/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 36/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 37/100
5815/5815 - 5s - loss: 0.3777 - accuracy: 0.8745
Epoch 38/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 39/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 40/100
5815/5815 - 5s - loss: 0.3777 - accuracy: 0.8745
Epoch 41/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 42/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 43/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 44/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 45/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 46/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 47/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 48/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 49/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 50/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 51/100
5815/5815 - 5s - loss: 0.3776 - accuracy: 0.8745
Epoch 52/100
```

Deep Model with 4 Layers

```
def create_larger():
```

```
  # create model
  model = Sequential()
  model.add(Dense(32, input_dim=26, activation='relu'))
  model.add(Dense(16, activation='relu'))
  model.add(Dense(8, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  # Compile model
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
```

```
# evaluate model with dataset
estimator = KerasClassifier(build_fn=create_larger, epochs=100, batch_size=5, verbose=2)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X_train, y_train, cv=kfold)
#print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
Epoch 22/100
4652/4652 - 5s - loss: 0.3755 - accuracy: 0.8757
Epoch 23/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 24/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 25/100
4652/4652 - 5s - loss: 0.3757 - accuracy: 0.8757
Epoch 26/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 27/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 28/100
4652/4652 - 5s - loss: 0.3755 - accuracy: 0.8757
Epoch 29/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 30/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 31/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 32/100
4652/4652 - 5s - loss: 0.3755 - accuracy: 0.8757
Epoch 33/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
```

```
Epoch 34/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 35/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 36/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 37/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 38/100
4652/4652 - 5s - loss: 0.3757 - accuracy: 0.8757
Epoch 39/100
4652/4652 - 5s - loss: 0.3757 - accuracy: 0.8757
Epoch 40/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 41/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 42/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 43/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 44/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 45/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 46/100
4652/4652 - 5s - loss: 0.3755 - accuracy: 0.8757
Epoch 47/100
4652/4652 - 5s - loss: 0.3755 - accuracy: 0.8757
Epoch 48/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 49/100
4652/4652 - 5s - loss: 0.3755 - accuracy: 0.8757
Epoch 50/100
4652/4652 - 5s - loss: 0.3756 - accuracy: 0.8757
Epoch 51/100
```

```
print("Deep MLP: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
Deep MLP: 87.57% (0.02%)
```

Final Model: We will be using `XGBoost` as our final model, as it performs substantially well compared to `Logistic Regression` & `MLP`