databricks Startup_Analysis_Part_One_Data_Preparation

Startup **A**nalysis

Part **O**ne

This project was conducted by Sascha Hagedorn, Maximilian Ott and Priya Matnani as part of the lecture IST 718 Advanced Information Analytics taught by Daniel Acuna at School of Information Studies, Syracuse University.

Project Abstract

The data was extracted from Crunchbase on February 2014. The dataset provides information about startup companies, investment, and acquisitions via Crunchbase.

Multiple supervised learning algorithms such as Logistic Regression, Random Forest and Neural Networks are applied after intense data preparation. Validation shows that Neural Networks has the best performance in this case.

With the results of this project existing startups can evaluate their performance in order to discover their probability to be acquired and emerging startups can use the outcomes as a guideline for how to structure their company or which features to

emphasize while going down the path of an emerging startup.

The data can be retrieved from: https://data.world/datanerd/startup-venture-funding (https://data.world/datanerd/startup-venture-funding)

Imports

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load functionality to manipulate dataframes
from pyspark.sql import functions as fn
from pyspark.sql.functions import col, monotonically_increasing_id, split,
when

# Functionality for computing features
from pyspark.ml import feature, regression, classification, Pipeline,
clustering
from pyspark.ml.classification import LogisticRegression,
RandomForestClassifier
from pyspark.ml.feature import RFormula, Tokenizer, VectorAssembler,
HashingTF, Word2Vec

from itertools import chain
from pyspark.ml.linalg import Vectors, VectorUDT

#Evaluation
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

Definition of custom functions

-- In this section custom functions are defined. This part is at the top to ensure, that "Run All" works properly. --

Custom functions are user defined functions. We defined them in order to do certain calculations for us. Each headline indicates to which procedure the functions belong.

functions for customized categories

```
#flatten list spark df to spark df
def flatten_list(df):
  test_pd = df.toPandas()
  mylist1 = []
  klist = []
  for i in range(len(test_pd.kmeans_feat)):
    newlist = list(chain.from_iterable(test_pd.category[i]))
    mylist1.append(newlist)
    klist.append(i)
  transferdf_pd = pd.DataFrame({"kmeans_feat": klist, "category": mylist1})
  newcatewoo = spark.createDataFrame(transferdf_pd)
  return newcatewoo


def three_highest_catpluscount(df):
  #to pandas
  test1 = df.select("tf").toPandas()
  maxi = []
  maxv = [0,0,0]
  indexlist = []
  valuelist = []
  kmeanlist = []
  for a in range(len(test1.tf)):
    for i in range(len(test1.tf[a])):
      if (min(maxv) < test1.tf[a][i]):
        maxv[0] = test1.tf[a][i]
        maxv = sorted(maxv)
    testarr = test1.tf[a].toArray()
    for n in range(2, -1, -1):
      index = np.where(testarr == maxv[n])
      maxi.append(index[0][0])
      valuelist.append(maxv[n])
      kmeanlist.append(a)
    maxv = [0,0,0]
  kmeancate_pd = pd.DataFrame({"categoryindex": maxi, "kmean_feat":
kmeanlist, "count": valuelist})
  kmeancate = spark.createDataFrame(kmeancate_pd)
  return kmeancate
```

```
# add categories to index
def add_category_words(df):
  kmeancate_pd = df.toPandas()
  mylisti = []
  for t in range(len(kmeancate_pd)):
    ind = kmeancate_pd.categoryindex[t]
    mylisti.append(wordvector_catwoo[ind])
  kmeancate_pd["category"] = mylisti
  kmeancate_words = spark.createDataFrame(kmeancate_pd)
  return kmeancate_words


def create_final_category(df):
  kmeancate_pd = df.toPandas()
  multiplecate = []
  categorylist = []
  kmeanlist = []
  for o in range(0,len(kmeancate_pd),3):
    multiplecate.append(kmeancate_pd["category"][o])
    if((kmeancate_pd["count"][o]*0.7 <= kmeancate_pd["count"][o+1]) and
(kmeancate_pd["category"][o] != kmeancate_pd["category"][o+1])):
      multiplecate.append(kmeancate_pd["category"][o+1])
      if((kmeancate_pd["count"][o+1]*0.7 <= kmeancate_pd["count"][o+2]) and
(kmeancate_pd["category"][o+1] != kmeancate_pd["category"][o+2])):
        multiplecate.append(kmeancate_pd["category"][o+2])
    categorystring = " + ".join(multiplecate)
    categorylist.append(categorystring)
    kmeanlist.append(o/3)
    multiplecate = []
  final_pd = pd.DataFrame({"category_final": categorylist, "kmean_feat":
kmeanlist})
  final = spark.createDataFrame(final_pd)
  return final
```

functions to get majority of investor country codes and funding round types

```
def blank_as_null(x):
    return when(col(x) != "", col(x)).otherwise(None)
```

```
def two_highest_catpluscount(df):
  #to pandas
  test1 = df.toPandas()
  maxi = []
  maxv = [0]
  indexlist = []
  valuelist = []
  kmeanlist = []
  for a in range(len(test1.tf)):
    for i in range(len(test1.tf[a])):
      if (min(maxv) < test1.tf[a][i]):
        maxv[0] = test1.tf[a][i]
        maxv = sorted(maxv)
    testarr = test1.tf[a].toArray()
    for n in range(1):
      index = np.where(testarr == maxv[n])
      maxi.append(index[0][0])
      valuelist.append(maxv[n])
      kmeanlist.append(test1.permalink[a])
    maxv = [0]
  kmeancate_pd = pd.DataFrame({"categoryindex": maxi, "perma": kmeanlist,
"count": valuelist})
  kmeancate = spark.createDataFrame(kmeancate_pd)
  return kmeancate


# add categories to index
def add_inv_words(df):
  kmeancate_pd = df.toPandas()
  mylisti = []
  for t in range(len(kmeancate_pd)):
    ind = kmeancate_pd.categoryindex[t]
    mylisti.append(inv_words[ind])
  kmeancate_pd["category"] = mylisti
  kmeancate_words = spark.createDataFrame(kmeancate_pd)
  return kmeancate_words


# add categories to index
def add_round_words(df):
  kmeancate_pd = df.toPandas()
  mylisti = []
  for t in range(len(kmeancate_pd)):
    ind = kmeancate_pd.categoryindex[t]
    mylisti.append(round_words[ind])
  kmeancate_pd["category"] = mylisti
  kmeancate_words = spark.createDataFrame(kmeancate_pd)
  return kmeancate_words
```

Get Data from Dropbox

To retrieve the data dynamically, the data is stored online at Dropbox. Thus, the path stays for all users the same and the data can be pulled easily. Github would have been a more sophistaced service for storing the data. Since the data exceeds 25MB, the files were to large for Github. The paths where the data can be found are listed below.

These paths are public and the data can be retrieved by anyone who has the link.

```
#pathnewacq = "https://www.dropbox.com/s/8xevxeuekw2mice/acquisitions.csv"
#pathnewadd = "https://www.dropbox.com/s/66g87yw6gw620y2/additions.csv"
#pathnewcom = "https://www.dropbox.com/s/d25fy3fp6bqjiid/companies.csv"
#pathnewinv = "https://www.dropbox.com/s/8cpf8osyy1hl9am/investments.csv"
#pathnewrou = "https://www.dropbox.com/s/neywvzrujmxykjr/rounds.csv"
```

get files to local server

```
%sh wget https://www.dropbox.com/s/dko78rtre7job62/acquisitions1.csv -nv
```

```
2018-01-17 21:05:08 URL:https://dl.dropboxusercontent.com/content_link/O9xbA
OuNexXcgv3mzx8t6A1MiPpbFesHWRL5wB1GlrCcgh87rTFY5GGvq8zGYs9G/file [3093963/30
93963] -> "acquisitions1.csv" [1]
```

```
%sh wget https://www.dropbox.com/s/66g87yw6gw620y2/additions.csv -nv
```

```
2018-01-17 21:05:09 URL:https://dl.dropboxusercontent.com/content_link/SwMWv
gGYm266Fdb0ax2FI9jc9r9LpK7yahjqtrX1bdJfE7iqTy5uoFt13ZM4XcU9/file [73703/7370
3] -> "additions.csv" [1]
```

```
%sh wget https://www.dropbox.com/s/d25fy3fp6bqjiid/companies.csv -nv
```

```
2018-01-17 21:05:11 URL:https://dl.dropboxusercontent.com/content_link/1Gafy
1j3jgzxlUulErNdtky27chBfKZz2YgzcAEPkXU0NyGLkp3fJnFb4IP5Pxnp/file [9810350/98
10350] -> "companies.csv" [1]
```

```
%sh wget https://www.dropbox.com/s/8cpf8osyy1hl9am/investments.csv -nv
```

```
2018-01-17 21:05:13 URL:https://dl.dropboxusercontent.com/content_link/FspdX
mHkfWWiYWaPQL4tPmVDgXK4vn44gZkjIYhrOXhfHOpOLOSl31VERNbQBBYS/file [33189750/3
3189750] -> "investments.csv" [1]
```

```
%sh wget https://www.dropbox.com/s/w3cjfl8v7cw3pcx/investments1.csv -nv
```

```
2018-01-17 21:05:18 URL:https://dl.dropboxusercontent.com/content_link/CysOw
e9Mg3T9IQIpsygtkaYyQGR30iAwFFs7RmOmJArXudCOWXHcbDYnSmZBR49a/file [32522147/3
2522147] -> "investments1.csv" [1]
```

```
%sh wget https://www.dropbox.com/s/neywvzrujmxykjr/rounds.csv -nv
```

```
2018-01-17 21:05:20 URL:https://dl.dropboxusercontent.com/content_link/25FDz
RxM8NHHmqoLTtcAKpASuQwyEWPikgtNYoq3ztQrkhU7LuyLYEr9iSV43mZD/file [17441018/1
7441018] -> "rounds.csv" [1]
```

load files into variables in the notebook

The data is loaded and the format .csv is specified. From now on, one can work with the data in the notebook.

```
dfacq =
sqlContext.read.format("csv").load("file:///databricks/driver/acquisitions1.
csv", delimiter = ",", header = True)
dfadd =
sqlContext.read.format("csv").load("file:///databricks/driver/additions.csv"
, delimiter = ",", header = True)
dfcom =
sqlContext.read.format("csv").load("file:///databricks/driver/companies.csv"
, delimiter = ",", header = True)
dfinv =
sqlContext.read.format("csv").load("file:///databricks/driver/investments1.c
sv", delimiter = ",", header = True)
dfrou =
sqlContext.read.format("csv").load("file:///databricks/driver/rounds.csv",
delimiter = ",", header = True)
```

clean errors in column names

After loading the data, we discovered that some column names have leading spaces, which leads to confusing column names since spaces in the beginning are hard to detect. This can easily cause errors and confusion. Hence, the column names are checked and the appropriate ones are renamed to a proper label. This procedure applies only to the Companies, the Acquisitions and the Rounds spreadsheets because the others did not have confusing column names.

```
dfcom = dfcom.select(col("permalink"), col("name"), col("homepage_url"),
col("category_list"), col(" market ").alias("market"), col("
funding_total_usd ").alias("funding_total_usd"), col("status"),
col("country_code"), col("state_code"), col("region"), col("city"),
col("funding_rounds"), col("founded_at"), col("founded_month"),
col("founded_quarter"), col("founded_year"), col("first_funding_at"),
col("last_funding_at"))
```

```
dfacq = dfacq.select(col("company_permalink"), col("company_name"),
col("company_category_list"), col("company_market"),
col("company_country_code"), col("company_state_code"),
col("company_region"), col("company_city"), col("acquirer_permalink"),
col("acquirer_name"), col("acquirer_category_list"), col("acquirer_market"),
col("acquirer_country_code"), col("acquirer_state_code"),
col("acquirer_region"), col("acquirer_city"), col("acquired_at"),
col("acquired_month"), col("acquired_quarter"), col("acquired_year"), col("
price_amount ").alias("price_amount"), col("price_currency_code"))
```

```
dfrou = dfrou.select(col("company_permalink"), col("company_name"),
col("company_category_list"), col("company_market"),
col("company_country_code"), col("company_state_code"),
col("company_region"), col("company_city"), col("funding_round_permalink"),
col("funding_round_type"), col("funding_round_code"), col("funded_at"),
col("funded_month"), col(" funded_quarter ").alias("funded_quarter"),
col("funded_year"), col(" raised_amount_usd ").alias("raised_amount_usd"))
```

Data Understanding

Data undestanding deals with a rather detailed investigation of the used data. We look for null values, existing dummy values and calculate basic statistics such as mean, max or min for numerical values. Categorical features and their number of unique categories are outlined. Moreover, different features and their occurring values are counted or summarized in order to get some first insights. Our examination is clustered into the 5 different spreadsheets.

Excel Spreadsheets with column names

- Companies
  - permalink

  - name
  - homepage_url
  - category_list
  - market
  - funding_total
  - status
  - country_code
  - state_code
  - region
  - city
  - funding_round
  - founded_at
  - founded_month
  - founded_quarter
  - founded_year
  - first_funding_at
  - last_funding_at
- Rounds
  - company_permalink
  - company_name
  - company_category_list
  - company_market
  - company_country_code
  - company_state_code
  - company_region
  - company_city
  - funding_round_peramlink
  - funding_round_type
  - funding_round_code
  - fundet_at
  - funded_month
  - funded_quarter
  - funded_year
  - raised_amount_usd
- Investements
  - company_permalink
  - company_name
  - company_category_list

  - company_market
  - company_country_code
  - company_state_code
  - company_region
  - company_city
  - investor_permalink
  - investor_name
  - investor_category_list
  - investor_market
  - investor_country_code
  - investor_region
  - investor_city
  - funding_round_permalink
  - funding_round_type
  - funding_round_code
  - funded_at
  - funded_month
  - funded_quarter
  - funded_year
  - rasied_amount_us
- Acquisitions
  - company_permalink
  - company_name
  - company_category_list
  - company_market
  - company_country_code
  - company_state_code
  - company_region
  - company_city
  - acquirer_permalink
  - acquirer_name
  - acquirer_category_list
  - acquirer_market
  - acquirer_country_code
  - acquirer_state_code
  - acquirer_region
  - acquirer_city
  - acquired_at

- acquired_month
- acquired_quarter
- acquired_year
- price_amount
- price_currency_code
- Additions
  - content
  - month_str
  - quarter_str
  - year_str
  - value

basic examination and statistics of companies data

```
display(dfcom)
```

permalink	name	homepage_url	category
/organization/waywire	#waywire	http://www.waywire.com	|Entertair
/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Games|
/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishir
/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electron Commer
/organization/r-ranch-and-mine	-R- Ranch and Mine	null	|Tourism|
/organization/club-domains	.Club Domains	http://nic.club/	|Software

Showing the first 1000 rows.

```
dfcom.count()
```

```
Out[22]: 49438
```

```
#show distribution of target variable
display(
  dfcom.select("status").\
  groupby(col("status")).\
  agg(fn.count("status"))
)
```

status
null
operating
acquired
closed

```
#check for null values in general and after that for each category of the
target variable
dfcom.toPandas().isnull().sum()
```

```
Out[24]:
permalink                0
name                     0
homepage_url          3449
category_list         3961
market                3968
funding_total_usd        0
status                1314
country_code          5273
state_code           19277
region                5273
city                  6116
funding_rounds           0
founded_at           10884
founded_month        10956
founded_quarter      10956
founded_year         10956
first_funding_at         0
last_funding_at          0
dtype: int64
```

```
dfcom.filter(col("status") == "closed").toPandas().isnull().sum()
```

```
Out[25]:
permalink               0
name                    0
homepage_url           31
category_list          78
market                 78
```

```
funding_total_usd       0
status                  0
country_code          408
state_code           1074
region                408
city                  441
funding_rounds          0
founded_at            607
founded_month         607
founded_quarter       607
founded_year          607
first_funding_at        0
last_funding_at         0
dtype: int64
```

```
dfcom.filter(col("status") == "operating").toPandas().isnull().sum()
```

```
Out[26]:
permalink                 0
name                      0
homepage_url           3092
category_list          3353
market                 3358
funding_total_usd         0
status                    0
country_code           4439
state_code            16696
region                 4439
city                   5166
funding_rounds            0
founded_at             9165
founded_month          9232
founded_quarter        9232
founded_year           9232
first_funding_at          0
last_funding_at           0
dtype: int64
```

```
dfcom.filter(col("status") == "acquired").toPandas().isnull().sum()
```

```
Out[27]:
permalink               0
name                    0
homepage_url          254
category_list         151
market                153
funding_total_usd       0
status                  0
country_code          220
state_code            804
region                220
```

```
city                    250
funding_rounds            0
founded_at              716
founded_month           721
founded_quarter         721
founded_year            721
first_funding_at          0
last_funding_at           0
dtype: int64
```

```
#get overview of different markets and their occurrence
markets = dfcom.select("market").\
  groupby(col("market")).\
  agg(fn.count("market")).\
  sort("count(market)", ascending=False)
```

```
display(markets.take(10))
```

market
Software
Biotechnology
Mobile
E-Commerce
Curated Web
Enterprise Software
Health Care
Clean Technology
Games

```
#get number of unique market categories
dfcom.select("market").distinct().count()
```

```
Out[30]: 754
```

```
#find highest funding_total_usd and appropriate startups
display(
  dfcom.select("permalink", "name", "funding_total_usd").\
  sort("funding_total_usd", ascending=False)
)
```

permalink
/organization/bayhill-therapeutics

/organization/victory-pharma
/organization/compstak
/organization/opendoor-2
/organization/quantisense
/organization/powercell-sweden
/organization/topcom-europe

Showing the first 1000 rows.

```
#understand where most of the startups are located - get cities with startup occurrence
cities = dfcom.select("permalink", "name", "city").\
  groupBy("city").\
  agg(fn.count("city")).\
  sort("count(city)", ascending=False)
```

```
display(cities.take(10))
```

city
San Francisco
New York
London
Palo Alto
Austin
Seattle
Cambridge
Chicago
Los Angeles

```
#understand where most of the startups are located - get countries with startup occurrence
countries = dfcom.select("permalink", "name", "country_code").\
  groupBy("country_code").\
  agg(fn.count("country_code")).\
  sort("count(country_code)", ascending=False)
```

```
display(countries)
```

country_code
USA
GBR
CAN
CHN
DEU
FRA
IND
ISR
ESP

```
display(
  dfcom.select("permalink", "name", "founded_year").\
  agg(fn.max("founded_year"), fn.min("founded_year"))
)
```

max(founded_year)
2014

```
#get basic statistics for columns with numerical value - understand min,
max, mean etc.
dfcom.select("funding_total_usd", "funding_rounds",
"founded_year").describe().show()
```

```
+-------+-----------------+------------------+-----------------+
|summary|funding_total_usd|    funding_rounds|     founded_year|
+-------+-----------------+------------------+-----------------+
|  count|            49438|             49438|            38482|
|   mean|329.3095238095238|1.6962053481127877|2007.359128943402|
| stddev|286.1580228709764| 1.294212699124526|7.579203055906465|
|    min|                -   |                 1|             1902|
|    max|        99,99,999 |                 9|             2014|
+-------+-----------------+------------------+-----------------+
```

basic examination and statistics of acquisitions data

```
display(dfacq)
```

company_permalink	company_name	company_category_list
/organization/waywire	#waywire	|Entertainment|Politics|Social Media|News|
/organization/fluff-friends	(fluff)Friends	null
/organization/red	(RED)	|Nonprofits|
/organization/vandaele-holdings	.	null

Showing the first 1000 rows.

```
dfacq.count()
```

```
Out[39]: 13070
```

```
#who are the companies who acquire the most
acquirers = dfacq.\
  select("acquirer_name").\
  groupby(col("acquirer_name")).\
  agg(fn.count("acquirer_name")).\
  sort("count(acquirer_name)", ascending=False)
```

```
display(acquirers.take(10))
```

acquirer_name
Cisco
Google
Microsoft
IBM
Yahoo!
Oracle Corporation

Hewlett-Packard
AOL

```
#statistics of price column
dfacq.select("price_amount").describe().show()
```

```
+-------+-------------------+
|summary|       price_amount|
+-------+-------------------+
|  count|               3716|
|   mean|7.553117933814094E8|
| stddev|4.082676696114348E9|
|    min|                -  |
|    max|          996000000|
+-------+-------------------+
```

```
#which different currencies occur
display(
  dfacq.\
  select("price_currency_code").\
  groupby("price_currency_code").\
  agg(fn.count("price_currency_code")).\
  sort("count(price_currency_code)", ascending=False)
)
```

price_currency_code
USD
EUR
GBP
CAD
AUD
JPY
SEK
NZD
SAR

basic examination and statistics of investment data

```
display(dfinv)
```

company_permalink	company_name	company_category_list	company
/organization/test-company-3	test company	null	null
/organization/andrewburnett-com-ltd	AndrewBurnett.com Ltd	|Internet|SEO|Services|Public Relations|Social Media|Consulting|	Internet
/organization/abo-data	ABO Data	|Enterprise Software|	Enterpris Software
/organization/abo-data	ABO Data	|Enterprise Software|	Enterpris Software
/organization/ikro	Ikro	null	null

Showing the first 1000 rows.

```
dfinv.count()
```

```
Out[45]: 114506
```

```
#check for null values
dfinv.toPandas().isnull().sum()
```

```
Out[46]:
company_permalink              0
company_name                   0
company_category_list       3264
company_market              3266
company_country_code        7359
company_state_code         35348
company_region              7359
company_city                8705
investor_permalink            66
investor_name                 66
investor_category_list     83999
investor_market            84051
investor_country_code      27985
investor_state_code        52232
investor_region            27985
investor_city              28499
funding_round_permalink        0
funding_round_type             0
funding_round_code         59837
funded_at                      0
funded_month                   0
funded_quarter                 0
```

```
funded_year                    0
raised_amount_usd          13351
dtype: int64
```

```
# get investor name, number of occurrences and total amount of invested usd
to detect top investors
investors = dfinv.\
  select("investor_name", "raised_amount_usd").\
  groupby(col("investor_name")).\
  agg(fn.count("investor_name"), fn.sum("raised_amount_usd")).\
  sort("count(investor_name)", ascending=False)
```

```
display(investors.take(7))
```

investor_name	count(invest
Sequoia Capital	776
Start-Up Chile	702
500 Startups	694
Intel Capital	674
Y Combinator	625
New Enterprise Associates	619
Accel Partners	592

```
#where are most of the investors located
investorcities = dfinv.\
  select("investor_city",).\
  groupby(col("investor_city")).\
  agg(fn.count("investor_city")).\
  sort("count(investor_city)", ascending=False)
```

```
display(investorcities.take(8))
```

investor_city
Menlo Park
New York
San Francisco
Palo Alto
London
Boston

Mountain View

```
#how many distinct investors do we have
dfinv.\
select("investor_permalink").\
distinct().\
count()
```

```
Out[51]: 22277
```

```
dfinv.select("raised_amount_usd", "funded_year").describe().show()
```

```
+-------+-------------------+------------------+
|summary|  raised_amount_usd|       funded_year|
+-------+-------------------+------------------+
|  count|             101155|            114506|
|   mean|1.335148888610089E7| 2010.686173650289|
| stddev|4.841612587381994E7|3.0655924875763825|
|    min|                 -  |              1921|
|    max|            9999997|              2014|
+-------+-------------------+------------------+
```

```
#how much money per year was invested over time
investperyear = dfinv.\
select("funded_year", "raised_amount_usd").\
groupby("funded_year").\
agg(fn.sum("raised_amount_usd")).\
sort("funded_year")
```

```
display(investperyear)
```

funded_year	sum(raised_ar
1921	null
1974	null
1979	2000000
1982	1044000
1983	null
1984	null
1985	1078000
1986	null
1987	2976000

basic examination and statistics of rounds data

```
display(dfrou)
```

company_permalink	company_name	company_category_list
/organization/waywire	#waywire	|Entertainment|Politics|Social Media|News|
/organization/tv-communications	&TV Communications	|Games|
/organization/tv-communications	&TV Communications	|Games|
/organization/rock-your-paper	'Rock' Your Paper	|Publishing|Education|
/organization/in-touch-network	(In)Touch Network	|Electronics|Guides|Coffee|Restaurants|Music|il Commerce|

Showing the first 1000 rows.

```
dfrou.count()
```

```
Out[56]: 83870
```

```
dfrou.toPandas().isnull().sum()
```

```
Out[57]:
company_permalink            0
company_name                 0
company_category_list     4446
company_market            4453
company_country_code      6566
company_state_code       27373
company_region            6566
company_city              7647
funding_round_permalink      0
funding_round_type           0
funding_round_code       61000
funded_at                    0
funded_month                10
funded_quarter              10
funded_year                 10
```

```
raised_amount_usd          12831
dtype: int64

#which funding round types happen more often than others - which are our
main types
fundingrounds = dfrou.\
  select("funding_round_type").\
  groupBy("funding_round_type").\
  agg(fn.count("funding_round_type")).\
  sort("count(funding_round_type)", ascending=False)
```

```
display(fundingrounds.take(7))
```

funding_round_type
venture
seed
debt_financing
angel
undisclosed
equity_crowdfunding
private_equity

```
dfrou.select("raised_amount_usd", "funded_year").describe().show()
```

```
+-------+-----------------+------------------+
|summary| raised_amount_usd|       funded_year|
+-------+-----------------+------------------+
|  count|            71039|             83860|
|   mean|299.36842105263156| 2011.018173145719|
| stddev| 270.4995795863479|2.8892397474174545|
|    min|             -   |              1921|
|    max|          99,999 |              2015|
+-------+-----------------+------------------+
```

Data Preparation

We discovered that the category column contains multiple labels concatenated by a bar. This would lead to many different category combinations, even though the main category would often be the same. Therefore, we decided to first cluster our different categories. After that, the most occurring single category in each cluster is extracted and used as new representative label for this cluster. This leads to a dimensionality reduction.

create custom category based on given category column

```
# Replace + with | , see Hardware+Software
df_categories_pd = dfcom.select("category_list").toPandas()
k = []
for i in df_categories_pd.category_list:
    i = str(i)
    j = i.replace(' + ','|')
    k.append(j)


# Create Spark DF with new Category list
mylist_pd_k = pd.DataFrame({"category_list": k})
df_cat_k = sqlContext.createDataFrame(mylist_pd_k)


#convert to Pandas
permalink_pd = dfcom.select("permalink").toPandas()


#create spark df of the categories as a list for each company
df_categories_pd = mylist_pd_k

mylist = []
for i in range(len(df_categories_pd)):
  mystring = str(df_categories_pd.category_list[i])
  myremover = mystring.split("|")
  str_list = filter(None, myremover)
  mylist.append(str_list)

mylist_pd = pd.DataFrame({"category": mylist, "permalink":
permalink_pd.permalink})
df_cat = sqlContext.createDataFrame(mylist_pd)


# counter vector + kmeans clustering + fitting and transforming
cv = feature.CountVectorizer(inputCol='category', outputCol='tf')
kmeans = clustering.KMeans(k=400, featuresCol='tf',
predictionCol='kmeans_feat')
pipeline_model = Pipeline(stages=[cv, kmeans]).fit(df_cat)
df_catid_kmeans = pipeline_model.transform(df_cat)
```

```
pipeline_model.stages[0].vocabulary
```

```
Out[66]:
[u'Software',
 u'Mobile',
 u'Biotechnology',
 u'None',
 u'E-Commerce',
 u'Curated Web',
 u'Social Media',
 u'Enterprise Software',
 u'Advertising',
 u'Games',
 u'Hardware',
 u'Health Care',
 u'Finance',
 u'Education',
 u'Clean Technology',
 u'Analytics',
 u'Health and Wellness',
 u'SaaS',
 u'Internet',
 u'Apps',
```

```
#display for checking results
#display(df_catid_kmeans.filter(col("kmeans_feat") == 0))
```

```
#concatenate the categories which are within one kmeans cluster
df_catid_kmeans_concat = df_catid_kmeans.\
groupby("kmeans_feat").\
agg(fn.collect_list(col("category")).alias("category")).\
sort("kmeans_feat")
```

```
#execute custom functions before the following

#flatten list spark df to spark df
newcatewoo = flatten_list(df_catid_kmeans_concat)
```

```
# counter vectorizer
cv = feature.CountVectorizer(inputCol='category', outputCol='tf')
```

```
# cv fitting and df transformation
cv_model = cv.fit(newcatewoo)
df_catwoo_cv = cv_model.transform(newcatewoo)
```

```
#get the three categories occuring most often for each cluster
kmeancate = three_highest_catpluscount(df_catwoo_cv)
```

```
wordvector_catwoo = cv_model.vocabulary
wordvector_catwoo
```

```
Out[73]:
[u'Software',
 u'Mobile',
 u'Biotechnology',
 u'None',
 u'E-Commerce',
 u'Curated Web',
 u'Social Media',
 u'Enterprise Software',
 u'Advertising',
 u'Games',
 u'Hardware',
 u'Health Care',
 u'Finance',
 u'Education',
 u'Clean Technology',
 u'Analytics',
 u'Health and Wellness',
 u'SaaS',
 u'Internet',
 u'Apps',
```

```
# add categories to index
kmeancate_words = add_category_words(kmeancate)
```

```
#create final representative category value for each cluster
final = create_final_category(kmeancate_words)
```

```
display(
  final.\
  groupby("category_final").\
  agg(fn.count("category_final").alias("count")).\
  filter(col("count") > 1).\
  sort(col("count").desc())
)
```

category_final
Software
Mobile
E-Commerce

Social Media
Curated Web
Enterprise Software
Advertising

```
#get number of unique categories
final.select("category_final").distinct().count()
```

```
Out[77]: 212
```

join spreadsheets

Since we have different spreadsheets, we decided to join spreadsheets to get one mastertable as basis for our analysis. Considering 'acquired' as target variable, only the companies and the investments spreadsheets contain significant information. Hence, those are joined. Since the investment spreadsheet can contain multiple investments for one company, the values of some features need to aggregated. While joining, we also calculated our own columns based on information of others.

check if companies are unique

```
#look for permalinks that occur twice
display(dfcom.select("permalink").groupBy("permalink").agg(fn.count("permali
nk")).where(fn.count("permalink") == 2))
```

permalink
/organization/prysm
/organization/treasure-valley-urology-services

```
#investigate duplicates
display(dfcom.filter((col("permalink") == "/organization/prysm")|
(col("permalink") == "/organization/treasure-valley-urology-services")))
```

permalink	name	homepage_url	category_list	market	func
/organization/prysm	Prysm	http://www.prysm.com/	null	null	14,
/organization/prysm	Prysm	http://www.prysm.com	|Displays|Hardware + Software|	Displays	14,
/organization/treasure-valley-urology-services	Treasure Valley Urology Services	null	|Biotechnology|	Biotechnology	2,8
/organization/treasure-valley-urology-services	Treasure Valley Urology	null	null	null	45,

remove duplicates with less information

```
#select observation with less information of duplicate 1
dfcomsubtr1 = dfcom.filter((col("permalink") == "/organization/prysm") &
((col("permalink") == "/organization/prysm") & (col("funding_rounds") ==
1)))


#select observation with less information of duplicate 2
dfcomsubtr2 = dfcom.filter((col("permalink") == "/organization/treasure-
valley-urology-services") & ((col("permalink") == "/organization/treasure-
valley-urology-services") & (col("funding_rounds") == 1)))


#get rid of duplicates (deprecated entries)
dfcom = dfcom.subtract(dfcomsubtr1).subtract(dfcomsubtr2)


#double check if preparation worked out
display(dfcom.select("permalink").groupBy("permalink").agg(fn.count("permali
nk")).sort("count(permalink)", ascending=False))
```

permalink
/organization/baanto-international
/organization/samares
/organization/windpipe
/organization/appsperse
/organization/snapsense
/organization/athoc

/organization/dna-health-corp

Showing the first 1000 rows.

define calculations for aggregation when joining the spreadsheets

```
#calculate the age of a company based on year of founding, the appropriate
quarter and 2015, which is the date when the data set was created
age_calc = fn.when(col("quarter_new") == "Q2", 2015 - col("founded_year") -
0.25).\
  otherwise(fn.when(col("quarter_new") == "Q3", 2015 - col("founded_year") -
0.5).\
            otherwise(fn.when(col("quarter_new") == "Q4", 2015 -
col("founded_year") - 0.75).\
                      otherwise(2015 - col("founded_year"))))


#calculate the time to funding of a company based on date of funding and
date of data set creation
time_to_funding_calc = fn.when(col("funded_quarter_new") == "Q2",
col("funded_year") + 0.25 - (2015 - col("age"))).\
  otherwise(fn.when(col("funded_quarter_new") == "Q3", col("funded_year") +
0.5 - (2015 - col("age"))).\
            otherwise(fn.when(col("funded_quarter_new") == "Q4",
col("funded_year") + 0.75 - (2015 - col("age"))).\
                      otherwise(col("funded_year") - (2015 - col("age")))))
```

define subsets before joining

```
#select subsets before joining to exclude information/observations which
will not be used
dfcomsub = dfcom.\
  withColumn("quarter_new", col("founded_quarter").substr(6,2)).\
  withColumn("age", age_calc).\
  select("permalink", "name", "market", "funding_total_usd", "status",
"country_code", "city", "funding_rounds", "founded_year",
"quarter_new","age")

dfinvsub = dfinv.\
  withColumn("funded_quarter_new", col("funded_quarter").substr(6,2)).\
  select("company_permalink", "investor_permalink", "investor_name",
"investor_country_code", "funding_round_type", "funded_quarter_new",
"funded_year", "raised_amount_usd")
```

join companies and investments data

```
#multiple rows in dfinv for each permalink
dfmaster = dfcomsub.join(dfinvsub, dfcomsub["permalink"] ==
dfinvsub["company_permalink"], 'leftouter')
dfmaster2 = dfmaster.\
  withColumn("time_to_funding", time_to_funding_calc)
```

aggregate multiple entries for each company

```
dfmaster2_agg = dfmaster2.\
  groupby(col("permalink").alias("permalink_agg")).\
  agg(fn.count("investor_permalink").alias("count_investor"),
      fn.min("time_to_funding").alias("time_to_first_funding"),
      fn.concat_ws(", ",
fn.collect_list(col("investor_country_code"))).alias("investor_country_codes
"),
      fn.concat_ws(", ",
fn.collect_list(col("funding_round_type"))).alias("funding_round_types"),
      fn.sum("raised_amount_usd").alias("total_raised_usd"))
```

```
dfmaster2_agg.count()
```

```
Out[89]: 49436
```

join aggregated values to companies subset dataframe

```
dfmaster_final = dfcomsub.join(dfmaster2_agg, dfcomsub["permalink"] ==
dfmaster2_agg["permalink_agg"], 'leftouter')
```

```
display(dfmaster_final)
```

permalink	name	market	funding_total_usd	status	c
/organization/1lay	1Lay	Mobile Security	1,70,000	operating	
/organization/24pagebooks	24PageBooks	Software	50,000	closed	
/organization/5min	5min Media	Video	1,28,00,000	acquired	
/organization/abpathfinder	ABPathfinder	Health and Wellness	9,60,000	operating	
/organization/acid-labs	Acid Labs	Software	-	operating	
/organization/aclaris-therapeutics	Aclaris Therapeutics	Biotechnology	4,20,00,000	operating	

Showing the first 1000 rows.

```
dfmaster_final.count()
```

```
Out[92]: 49436
```

```
#examination of observations with target variable acquired
display(dfmaster.where(col("status") == "acquired"))
```

permalink	name	market	funding_total_usd	status	country_co
/organization/5min	5min Media	Video	1,28,00,000	acquired	USA
/organization/5min	5min Media	Video	1,28,00,000	acquired	USA
/organization/5min	5min Media	Video	1,28,00,000	acquired	USA
/organization/adaptivity	Adaptivity	Enterprise Software	2,48,45,955	acquired	USA

Showing the first 1000 rows.

```
#drop column which exists twice with different name
dfmaster3 = dfmaster2.drop("company_permalink")
```

investigate quarter of founding and funds

```
#summarize raised amount of USD for each quarter
investperquarter = dfmaster3.\
select("quarter_new", "raised_amount_usd").\
groupby("quarter_new").\
agg(fn.sum("raised_amount_usd")).\
sort("quarter_new")
```

```
investperquarternona = investperquarter.na.drop()
```

```
#companies which were founded in quarter 1 tend to get more funds than
others
display(investperquarternona)
```

quarter_new	sum(raised_a
Q1	83162948607(
Q2	84458713629
Q3	96481223209
Q4	11655549748{

join custom category column to dataframe of companies and investments data

```
#select permalink, category and kmeans_feat
df_join1 = df_catid_kmeans.\
  select("permalink", "category", "kmeans_feat")
```

```
#join to final custom category df
df_final_permas = df_join1.join(final, df_join1["kmeans_feat"] ==
final["kmean_feat"], 'leftouter')
```

```
display(df_final_permas)
```

permalink	category
/organization/1-800-dentist	▸ ["Health and Wellness"]
/organization/1-800-doctors	▸ ["Health and Wellness"]
/organization/fitfrnd-2	▸ ["Personal Health","Health and Wellness
/organization/1eq	▸ ["Mobile Health","Health and Wellness"]
/organization/21st-century-oncology	▸ ["Health and Wellness"]
/organization/39-health	▸ ["Health and Wellness"]
/organization/720	▸ ["Predictive Analytics","Analytics","Health
/organization/80th-street-residence-facc-fund-i	▸ ["Health and Wellness"]
/organization/a-better-tomorrow-treatment-center	▸ ["Health and Wellness"]

Showing the first 1000 rows.

```
#select subset and rename permalink column to ensure unique column names
df_final_permas_sub = df_final_permas.\
  select(col("permalink").alias("permalink_sub"),"category_final")
```

```
#join custom categories to master df which is the one resulting from joining
companies and investment spreadsheet (inlcuding aggregation)
df_master_final_cate = dfmaster_final.join(df_final_permas_sub,
dfmaster_final["permalink"] == df_final_permas_sub["permalink_sub"],
'leftouter')
```

```
display(df_master_final_cate)
```

permalink	name	market	funding_total_usd	status	c
/organization/1lay	1Lay	Mobile Security	1,70,000	operating	
/organization/24pagebooks	24PageBooks	Software	50,000	closed	
/organization/5min	5min Media	Video	1,28,00,000	acquired	
/organization/abpathfinder	ABPathfinder	Health and Wellness	9,60,000	operating	

/organization/acid-labs	Acid Labs	Software	-	operating

Showing the first 1000 rows.

get majority of investor_country_codes and funding_round_types to reduce complexity

Since a company could have had multiple investors from different countries and could have faced multiple different funding rounds, the aggregated data can contain a list of the different country codes of the investors or a list with different fuding round types. This leads to many different categories. To reduce dimension and complexity we decided to look for the most occurring investor country code or funding round type of each company and to use this value as representative label.

```
#select subset of appropriate columns
dfmastermajority = df_master_final_cate.select("permalink",
"investor_country_codes", "funding_round_types")


#the blank strings have to get converted to NULL for further functions
dfmajority = dfmastermajority.\
  withColumn("investor_country_codes",
blank_as_null("investor_country_codes")).\
  withColumn("funding_round_types", blank_as_null("funding_round_types"))


#write investor_country_codes string and funding_round_types strings that a
separated by commas into a list
majority = dfmajority.\
  withColumn("investor_country_codes", split(col("investor_country_codes"),
",\s*")).\
  withColumn("funding_round_types", split(col("funding_round_types"),
",\s*"))


#dropping of rows with NULL values
majoritydropinv =
majority.select("permalink","investor_country_codes").na.drop()
majoritydropround =
majority.select("permalink","funding_round_types").na.drop()
```

```
# Counter vectorizing the investor_country_codes and funding_round_types
feature in order to use the vector to calculate the majority count for each
observation

cv_inv = feature.CountVectorizer(inputCol='investor_country_codes',
outputCol='tf')

cv_round = feature.CountVectorizer(inputCol='funding_round_types',
outputCol='tf')


cv_inv_model = cv_inv.fit(majoritydropinv)
df_cv_inv = cv_inv_model.transform(majoritydropinv)

cv_round_model = cv_round.fit(majoritydropround)
df_cv_round = cv_round_model.transform(majoritydropround)


#calculate the two highest counts of the investor_country_code for each
company
invtwo = two_highest_catpluscount(df_cv_inv)


#assign the words of the counter vectorizer of the investor_country_codes
inv_words = cv_inv_model.vocabulary


#assign the corresponding investor_country_codes to the counts for each
company
invplus_words = add_inv_words(invtwo)


#calculate the two highest counts of the funding_round_types for each
company
roundtwo = two_highest_catpluscount(df_cv_round)


#assign the corresponding funding_round_types to the counts for each company
round_words = cv_round_model.vocabulary


#assign the corresponding funding_round_types to the counts for each company
roundpluswords = add_round_words(roundtwo)


#join the "majority" investor_country_codes to the master table
masternew =
df_master_final_cate.join(invplus_words.select("perma",col("category").alias
("investor_country_code")), df_master_final_cate["permalink"] ==
invplus_words["perma"], 'leftouter')
```

```
#selection in order to rename and for better joining
roundpluswords =
roundpluswords.select(col("perma").alias("permaround"),col("category").alias
("funding_round_type"))
```

```
#join the "majority" funding_round_types to the master table
masternew = masternew.join(roundpluswords, masternew["permalink"] ==
roundpluswords["permaround"], 'leftouter')
```

```
display(masternew)
```

permalink	name	market	funding_total_usd	status	c
/organization/1lay	1Lay	Mobile Security	1,70,000	operating	
/organization/24pagebooks	24PageBooks	Software	50,000	closed	
/organization/5min	5min Media	Video	1,28,00,000	acquired	
/organization/abpathfinder	ABPathfinder	Health and Wellness	9,60,000	operating	
/organization/acid-labs	Acid Labs	Software	-	operating	
/organization/aclaris-therapeutics	Aclaris Therapeutics	Biotechnology	4,20,00,000	operating	

Showing the first 1000 rows.

```
#this dataframe was exported and then imported in another notebook
#the entire project is split into two noteboos

masterdropped = masternew.drop("funding_total_usd", "permalink_agg",
"investor_country_codes", "funding_round_types", "permalink_sub", "perma",
"permaround")
```

```
masterdropped.count()

Out[120]: 49444
```

This dataframe called "masternew" is exported and then imported into Notebook 2.

```
#display(masterdropped)
```

Some additional analysis (basic statistics) of the data in masternew follows here.

create binary value for target variable

Since a classification predicts if a certain label case is true or not, the target variable needs to have a 1 or 0 representation.

```
#create new column with 1 or 0 depending on value of target variable:
acquired = 1, otherwise 0
finaltarget = masterdropped.\
  withColumn("label", fn.when(col("status") == "acquired",1).otherwise(0))
```

```
display(finaltarget)
```

permalink	name	market	status	country_code	city
/organization/1lay	1Lay	Mobile Security	operating	null	null
/organization/24pagebooks	24PageBooks	Software	closed	USA	Roch
/organization/5min	5min Media	Video	acquired	USA	New
/organization/abpathfinder	ABPathfinder	Health and Wellness	operating	USA	Over Park
/organization/acid-labs	Acid Labs	Software	operating	USA	Santa Moni
/organization/aclaris-therapeutics	Aclaris Therapeutics	Biotechnology	operating	USA	Malv

Showing the first 1000 rows.

drop missing values

```
#create subset of dataframe without missing values
finalwithoutna = finaltarget.na.drop()
```

Understanding of final dataset without missing values

The final mastertable is also examined, since data preparations added new columns and missing values are dropped.

```
df = finalwithoutna
```

```
#distribution of categories in target variable
display(
  df.select("status").\
  groupby(col("status")).\
  agg(fn.count("status"))
)
```

status
operating
acquired
closed

```
#class balance target label
display(
  df.select("label").\
  groupby(col("label")).\
  agg(fn.count("label"))
)
```

label	count(label)
1	2079
0	12681

```
#which categories occur most often
categories_master = df.select("category_final").\
  groupby(col("category_final")).\
  agg(fn.count("category_final")).\
  sort("count(category_final)", ascending=False)
```

```
display(categories_master.take(10))
```

category_final
Software
Biotechnology
Enterprise Software
Mobile
E-Commerce
Curated Web
Advertising
Analytics
Software + Hardware

```
#investigate in which cities most of the startups are
cities_master = df.select("name", "city").\
  groupBy("city").\
  agg(fn.count("city")).\
  sort("count(city)", ascending=False)
```

```
display(cities_master.take(10))
```

city
San Francisco
New York
London
Palo Alto
Mountain View
Cambridge
Seattle
Austin
Boston

```
#investigate in which countries most of the startups of our data are
countries_master = df.select("name", "country_code").\
  groupBy("country_code").\
  agg(fn.count("country_code")).\
  sort("count(country_code)", ascending=False)
```

```
display(countries_master)
```

country_code
USA
GBR
CAN
CHN
FRA
ISR
DEU
IND
ESP

```
# get compnay name, number of investment and total amount of invested usd
sorted by number of investors
investors_master_count = df.\
  select("name", "count_investor", "total_raised_usd", "category_final").\
  groupby(col("name")).\
  agg(fn.max("count_investor"), fn.max("total_raised_usd"),
fn.first("category_final")).\
  sort("max(count_investor)", ascending=False)
```

```
display(investors_master_count.take(10))
```

name	max(count_investor)
Fab	60
ecomom	59
CardioDx	57
Practice Fusion	55
Path	53
Aperto Networks	49

EndoGastric Solutions	49

```
# get investor name, number of investment and total amount of invested usd
sorted by raised usd
investors_master_usd = df.\
  select("name", "count_investor", "total_raised_usd", "category_final").\
  groupby(col("name")).\
  agg(fn.max("count_investor"), fn.max("total_raised_usd"),
fn.first("category_final")).\
  sort("max(total_raised_usd)", ascending=False)
```

```
display(investors_master_usd.take(10))
```

name	max(count_investor)
Clearwire	17
Flipkart	32
Groupon	23
Uber	40
Venari Resources	4
Nanosolar	36
Dropbox	23
Facebook	20
Pinterest	42

```
# get categories of funding rounds and the number of their occurrence
fundingrounds_master = df.\
  select("funding_round_type").\
  groupBy("funding_round_type").\
  agg(fn.count("funding_round_type")).\
  sort("count(funding_round_type)", ascending=False)
```

```
display(fundingrounds_master.take(5))
```

funding_round_type
venture
seed

angel
private_equity
undisclosed

End of notebook 1

We decided to split our notebook, because it became too long and we experienced some delay when scrolling or running code. The dataframe called "masternew" was exported, uploaded to Dropbox and will be imported at the very beginning of **Notebook 2**.