Import required packages and libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

# plot graph
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

# packages for models
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import *

import warnings
warnings.filterwarnings('ignore')
```

Load dataset

In [2]:

```
df = pd.read_csv('Investments.csv',on_bad_lines='skip',encoding = "ISO-8859-1")
```

In [3]:

```
df
```

Out[3]:

	permalink	name	homepage_url	category_list	market	funding_total_
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|News|	News	17,50,
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Games|	Games	40,00,
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Education|	Publishing	40,
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Music|i...	Electronics	15,00,
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Games|	Tourism	60,
...	...	...	...	...	...	
54289	NaN	NaN	NaN	NaN	NaN	N
54290	NaN	NaN	NaN	NaN	NaN	N
54291	NaN	NaN	NaN	NaN	NaN	N
54292	NaN	NaN	NaN	NaN	NaN	N
54293	NaN	NaN	NaN	NaN	NaN	N

54294 rows × 39 columns

In [4]:

```
df.head()
```

Out[4]:

	permalink	name	homepage_url	category_list	market	funding_total_usd
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|News|	News	17,50,000
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Games|	Games	40,00,000
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Education|	Publishing	40,000
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Music|i...	Electronics	15,00,000
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Games|	Tourism	60,000

5 rows × 39 columns

Data Cleaning

In [5]:

```
len(df)
```

Out[5]:

```
54294
```

In [6]:

```
df.tail()
```

Out[6]:

	permalink	name	homepage_url	category_list	market	funding_total_usd	status	country_code	state_code	region	...	secondary_mark
54289	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
54290	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
54291	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
54292	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
54293	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na

5 rows × 39 columns

as you can see in two outputs above, we have 54,294 rows of data but some them not contain any information. It may lead to misdirection summary when we do some analysis or visualize them. Then, I just remove them by select only data which has "name column"

In [7]:

```
# select only data which name is not null

df = df[~df.name.isna()]
```

In [8]:

```
df = df[~df.name.isna()]
len(df)
```

Out[8]:

```
49437
```

In [9]:

```
df.name.isna()
```

Out[9]:

```
0        False
1        False
2        False
3        False
4        False
         ...
49433    False
49434    False
49435    False
49436    False
49437    False
Name: name, Length: 49437, dtype: bool
```

In [10]:

```
len(df)
```

Out[10]:

```
49437
```

we have around 49,437 companies left in our dataset

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49437 entries, 0 to 49437
Data columns (total 39 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   permalink             49437 non-null  object
 1   name                  49437 non-null  object
 2   homepage_url          45988 non-null  object
```

```
 3   category_list         45476 non-null  object
 4    market               45469 non-null  object
 5    funding_total_usd    49437 non-null  object
 6   status                48123 non-null  object
 7   country_code          44165 non-null  object
 8   state_code            30161 non-null  object
 9   region                44165 non-null  object
 10  city                  43322 non-null  object
 11  funding_rounds        49437 non-null  float64
 12  founded_at            38553 non-null  object
 13  founded_month         38481 non-null  object
 14  founded_quarter       38481 non-null  object
 15  founded_year          38481 non-null  float64
 16  first_funding_at      49437 non-null  object
 17  last_funding_at       49437 non-null  object
 18  seed                  49437 non-null  float64
 19  venture               49437 non-null  float64
 20  equity_crowdfunding   49437 non-null  float64
 21  undisclosed           49437 non-null  float64
 22  convertible_note      49437 non-null  float64
 23  debt_financing        49437 non-null  float64
 24  angel                 49437 non-null  float64
 25  grant                 49437 non-null  float64
 26  private_equity        49437 non-null  float64
 27  post_ipo_equity       49437 non-null  float64
 28  post_ipo_debt         49437 non-null  float64
 29  secondary_market      49437 non-null  float64
 30  product_crowdfunding  49437 non-null  float64
 31  round_A               49437 non-null  float64
 32  round_B               49437 non-null  float64
 33  round_C               49437 non-null  float64
 34  round_D               49437 non-null  float64
 35  round_E               49437 non-null  float64
 36  round_F               49437 non-null  float64
 37  round_G               49437 non-null  float64
 38  round_H               49437 non-null  float64
dtypes: float64(23), object(16)
memory usage: 15.1+ MB
```

In [12]:

```
df.head()
```

Out[12]:

	permalink	name	homepage_url	category_list	market	funding_total_usd
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|News|	News	17,50,000
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Games|	Games	40,00,000
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Education|	Publishing	40,000
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Music|i...	Electronics	15,00,000
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Games|	Tourism	60,000

5 rows × 39 columns

Deal with NaN values and Encode Categorical Features

In [13]:

```
df.isna().sum()
```

Out[13]:

```
permalink                 0
name                      0
homepage_url           3449
category_list          3961
 market                3968
 funding_total_usd        0
status                 1314
country_code           5272
state_code            19276
region                 5272
city                   6115
funding_rounds            0
founded_at            10884
founded_month         10956
founded_quarter       10956
founded_year          10956
```

```
first_funding_at           0
last_funding_at            0
seed                       0
venture                    0
equity_crowdfunding        0
undisclosed                0
convertible_note           0
debt_financing             0
angel                      0
grant                      0
private_equity             0
post_ipo_equity            0
post_ipo_debt              0
secondary_market           0
product_crowdfunding       0
round_A                    0
round_B                    0
round_C                    0
round_D                    0
round_E                    0
round_F                    0
round_G                    0
round_H                    0
dtype: int64
```

In [14]:

```
print(df.columns.values)
```

```
['permalink' 'name' 'homepage_url' 'category_list' ' market '
 ' funding_total_usd ' 'status' 'country_code' 'state_code' 'region'
 'city' 'funding_rounds' 'founded_at' 'founded_month' 'founded_quarter'
 'founded_year' 'first_funding_at' 'last_funding_at' 'seed' 'venture'
 'equity_crowdfunding' 'undisclosed' 'convertible_note' 'debt_financing'
 'angel' 'grant' 'private_equity' 'post_ipo_equity' 'post_ipo_debt'
 'secondary_market' 'product_crowdfunding' 'round_A' 'round_B' 'round_C'
 'round_D' 'round_E' 'round_F' 'round_G' 'round_H']
```

From those columns above, I will select "status","market","funding total usd","founded at","country code"to do some roughly describe the dataset. warning some column name contains sapce in string, I decide to remove them first.

In [15]:

```
df.rename(columns={'funding_total_usd':"funding_total_usd",
                   'market': "market"},inplace=True)
```

double click (or enter) to edit status

In [16]:

```
df["status"].value_counts()
```

Out[16]:

```
operating    41829
acquired      3692
closed        2602
Name: status, dtype: int64
```

In [17]:

```
df.describe()
```

Out[17]:

	funding_rounds	founded_year	seed	venture	equity_crowdfunding	undisclosed	convertible_note	debt_financing	
count	49437.000000	38481.000000	4.943700e+04	4.943700e+04	4.943700e+04	4.943700e+04	4.943700e+04	4.943700e+04	4.94370
mean	1.696219	2007.359034	2.173254e+05	7.501202e+06	6.163447e+03	1.302239e+05	2.336457e+04	1.888195e+06	6.54203
std	1.294222	7.579279	1.056995e+06	2.847139e+07	1.999068e+05	2.981434e+06	1.432060e+06	1.382060e+08	6.58297
min	1.000000	1902.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00000
25%	1.000000	2006.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00000
50%	1.000000	2010.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00000
75%	2.000000	2012.000000	2.500000e+04	5.000000e+06	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.00000
max	18.000000	2014.000000	1.300000e+08	2.351000e+09	2.500000e+07	2.924328e+08	3.000000e+08	3.007950e+10	6.35902

8 rows × 23 columns

In [18]:

```
# describing a non numerical data
df.describe(exclude=[np.number])
```

Out[18]:

	permalink	name	homepage_url	category_list	market	funding_total_usd	status	country_code	state_code	regic

count	49437	49437	45988	45476	45469	49437	48123	44165	30161	4416
unique	49435	49350	45849	16675	753	14617	3	115	61	108
top	/organization/prysm	Roost	http://www.smartfocus.com	|Software|	Software	-	operating	USA	CA	S Ba Are
freq	2	4	2	3650	4620	8531	41829	28793	9917	680

In [19]:

```
df.region
```

Out[19]:

```
0        New York City
1          Los Angeles
2              Tallinn
3               London
4               Dallas
             ...
49433           London
49434          Beijing
49435            Split
49436              NaN
49437    New York City
Name: region, Length: 49437, dtype: object
```

In [20]:

```
df.city == "New York"
```

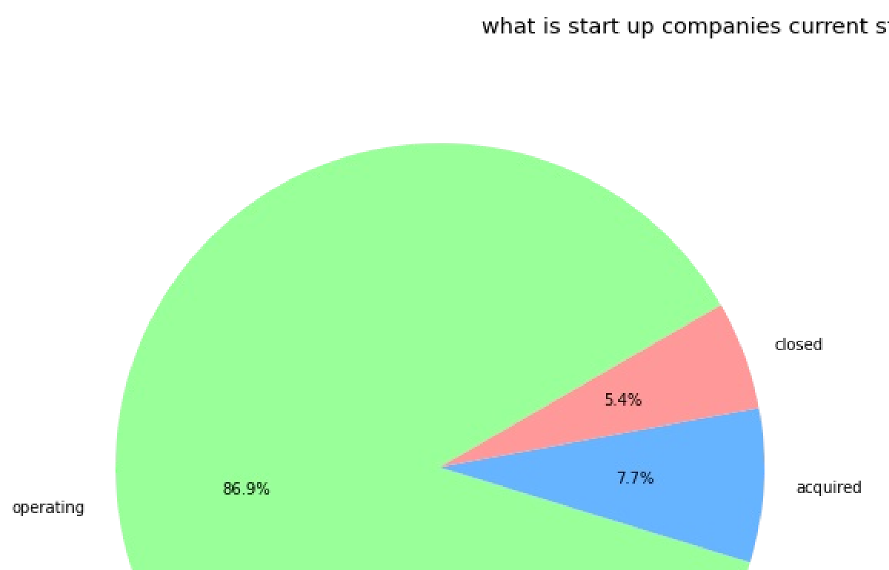
Out[20]:

```
0         True
1        False
2        False
3        False
4        False
         ...
49433    False
49434    False
49435    False
49436    False
49437     True
Name: city, Length: 49437, dtype: bool
```

visualization

In [21]:

```
plt.rcParams['figure.figsize'] = 8,8
labels = df['status'].value_counts().index.tolist()
sizes = df['status'].value_counts().tolist()
explode = (0, 0, 0.2)
colors = ['#99ff99','#66b3ff','#ff9999']


plt.pie(sizes, explode=None, labels=labels, colors=colors, autopct='%1.1f%%', shadow=False, startangle=30)
plt.axis('equal')
plt.tight_layout()
plt.title("what is start up companies current status", fontdict=None, position= [0.88,2], size = 'x-large')
plt.show()
```



Most of company (86.9%) in the dataset is operating, and around 5.4% company is already closed.

In [22]:
```
df.columns = [i.strip() for i in df.columns]   # remove the extra space in market column
```

In [23]:
```
df.columns
```

Out[23]:
```
Index(['permalink', 'name', 'homepage_url', 'category_list', 'market',
       'funding_total_usd', 'status', 'country_code', 'state_code', 'region',
       'city', 'funding_rounds', 'founded_at', 'founded_month',
       'founded_quarter', 'founded_year', 'first_funding_at',
       'last_funding_at', 'seed', 'venture', 'equity_crowdfunding',
       'undisclosed', 'convertible_note', 'debt_financing', 'angel', 'grant',
       'private_equity', 'post_ipo_equity', 'post_ipo_debt',
       'secondary_market', 'product_crowdfunding', 'round_A', 'round_B',
       'round_C', 'round_D', 'round_E', 'round_F', 'round_G', 'round_H'],
      dtype='object')
```

In [24]:
```
len('market')
```

Out[24]:
```
6
```

In [25]:
```
len(df['market'].unique())
```

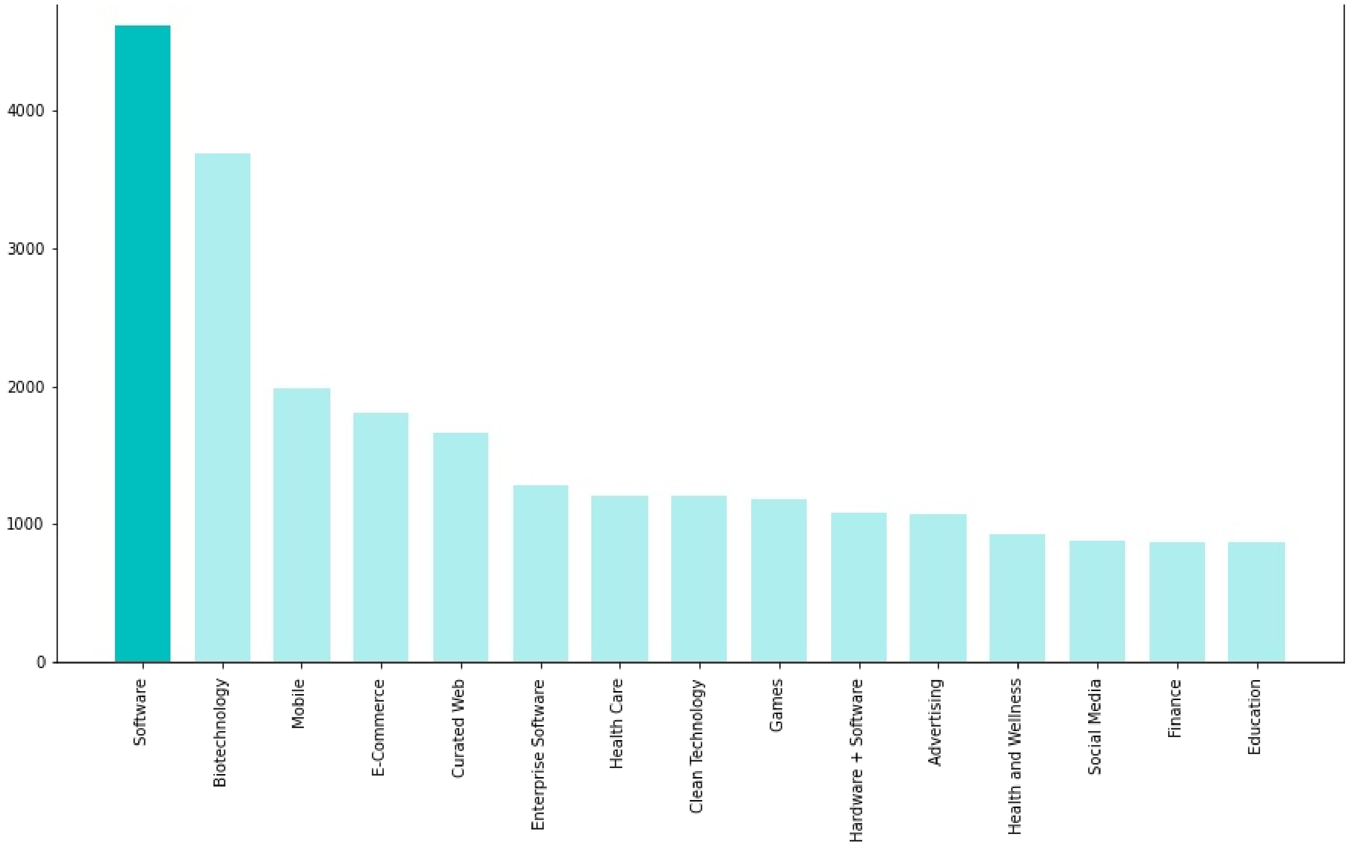
Out[25]:
```
754
```

In [26]:
```
df['market'].value_counts()[:15]
```

Out[26]:
```
 Software                 4620
 Biotechnology            3688
 Mobile                   1983
 E-Commerce               1805
 Curated Web              1655
 Enterprise Software      1280
 Health Care              1207
 Clean Technology         1200
 Games                    1182
 Hardware + Software      1081
 Advertising              1064
 Health and Wellness       920
 Social Media              876
 Finance                   867
 Education                 863
Name: market, dtype: int64
```

because we have around 754 categories of start up, then just plot the top 15

In [28]:
```
plt.rcParams['figure.figsize'] = 15,8

height = df['market'].value_counts()[:15].tolist()
bars = df['market'].value_counts()[:15].index.tolist()
y_pos = np.arange(len(bars))
plt.bar(y_pos, height , width=0.7 , color=['c']+['paleturquoise']*14)
plt.xticks(y_pos, bars)
plt.xticks(rotation=90)
plt.title("Top 15 start-Up Market Category", fontdict=None, position= [0.48,1.05], size = 'x-large')
plt.show()
```



Category

In [29]:

```
df.category_list
```

Out[29]:

```
0                |Entertainment|Politics|Social Media|News|
1                                                   |Games|
2                                    |Publishing|Education|
3        |Electronics|Guides|Coffee|Restaurants|Music|i...
4                             |Tourism|Entertainment|Games|
                               ...
49433    |Analytics|Gamification|Developer APIs|iOS|And...
49434                                 |Enterprise Software|
49435         |Web Development|Advertising|Wireless|Mobile|
49436                                               |Games|
49437                                 |Enterprise Software|
Name: category_list, Length: 49437, dtype: object
```

In [31]:

```
set_keywords = set()
for liste_keywords in df['category_list'].str.split('|').values:
    if isinstance(liste_keywords, float): continue  #only happen if lists_keywords = NaN
    set_keywords = set_keywords.union(liste_keywords)
#_____________________________
#remove null chain entry
set_keywords.remove('')
```

The most popular category is still about software & mobile, it maybe because these 2 categories are easily to scalable?

Total Funding USD

In [33]:

```
df['funding_total_usd'].head()
```

Out[33]:

```
0     17,50,000
1     40,00,000
2        40,000
3     15,00,000
4        60,000
Name: funding_total_usd, dtype: object
```

Unlucky...., this column is provided in 'string' format which contain comma(,) minus(-) and space() we need to repace them and covert it to numeric format first

In [38]:
```python
df['funding_total_usd'] = df['funding_total_usd'].str.replace(',', '')
df['funding_total_usd'] = df['funding_total_usd'].str.replace('-', '')
df['funding_total_usd'] = df['funding_total_usd'].str.replace(' ', '')

df['funding_total_usd'] = pd.to_numeric(df['funding_total_usd'], errors='coerce')
```

In [39]:
```python
df['funding_total_usd'].head()
```

Out[39]:
```
0    1750000.0
1    4000000.0
2      40000.0
3    1500000.0
4      60000.0
Name: funding_total_usd, dtype: float64
```

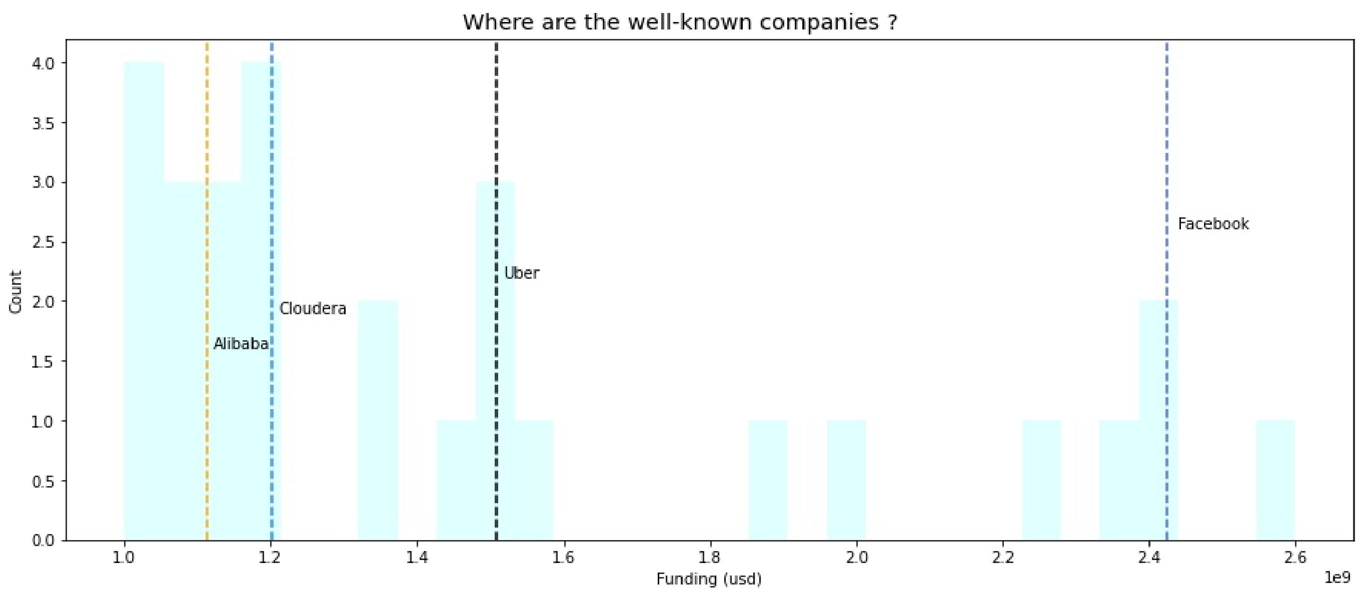
okay... let do some visualize Seem like it has large gap between the highest value and the lowest,let ignore outlier first:) I will use the simple remove outlier technique such as 1.5IQR

In [42]:
```python
Q1 = df['funding_total_usd'].quantile(0.25)
Q3 = df['funding_total_usd'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = (Q1 - 1.5 * IQR)
upper_bound = (Q3 + 1.5 * IQR)
```

In [43]:
```python
without_outlier = df[(df['funding_total_usd'] > lower_bound) & (df['funding_total_usd'] < upper_bound)]
```

In [45]:
```python
Facebook_total_funding = df['funding_total_usd'][df['name']=="Facebook"].values[0]
Uber_total_funding = df['funding_total_usd'][df['name']=="Uber"].values[0]
Alibaba_total_funding = df['funding_total_usd'][df['name']=="Alibaba"].values[0]
Cloudera_total_funding = df['funding_total_usd'][df['name']=="Cloudera"].values[0]
```

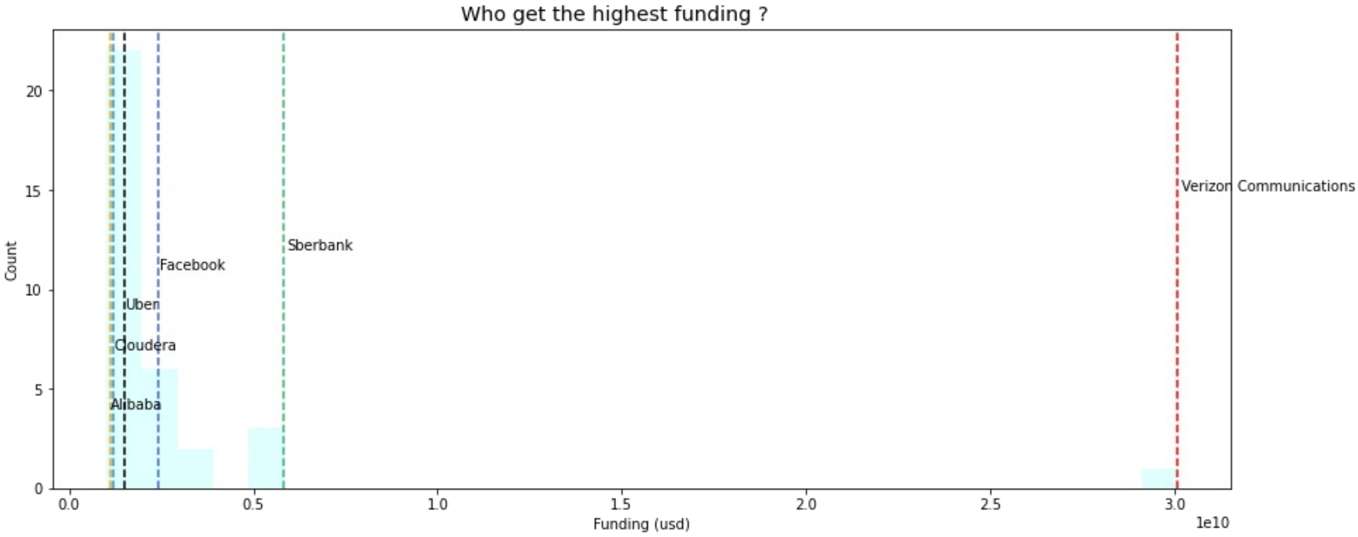
In [46]:
```python
plt.rcParams['figure.figsize'] = 15,6

plt.hist(df['funding_total_usd'][(df['funding_total_usd'] >= 1000000000)&(df['funding_total_usd'] <= 3000000000)]
plt.ylabel('Count')
plt.xlabel('Funding (usd)')
plt.title("Where are the well-known companies ? ", fontdict=None, position= [0.48,1.05], size = 'x-large')

plt.axvline(Facebook_total_funding,color='royalblue',linestyle ="--")
plt.text(Facebook_total_funding+15000000, 2.6,"Facebook")

plt.axvline(Uber_total_funding,color='black',linestyle ="--")
plt.text(Uber_total_funding+10000000, 2.2,"Uber")

plt.axvline(Cloudera_total_funding,color='dodgerblue',linestyle ="--")
plt.text(Cloudera_total_funding+10000000, 1.9,"Cloudera")

plt.axvline(Alibaba_total_funding,color='orange',linestyle ="--")
plt.text(Alibaba_total_funding+10000000, 1.6,"Alibaba")




plt.show()
```



But..., Are they the highest funding ? the answer is no.

In [47]:
```
Verizon_total_funding = df['funding_total_usd'][df['name']=="Verizon Communications"].values[0]
Sberbank_total_funding = df['funding_total_usd'][df['name']=="Sberbank"].values[0]
```

In [48]:
```
plt.rcParams['figure.figsize'] = 15,6
plt.hist(df['funding_total_usd'][(df['funding_total_usd'] >= 1000000000)].dropna(), bins=30,color = 'lightcyan' )
plt.ylabel('Count')
plt.xlabel('Funding (usd)')
plt.title("Who get the highest funding ? ", fontdict=None, position= [0.48,1.05], size = 'x-large')

plt.axvline(Facebook_total_funding,color='royalblue',linestyle ="--")
plt.text(Facebook_total_funding+15000000, 11,"Facebook")

plt.axvline(Uber_total_funding,color='black',linestyle ="--")
plt.text(Uber_total_funding+10000000, 9,"Uber")

plt.axvline(Cloudera_total_funding,color='dodgerblue',linestyle ="--")
plt.text(Cloudera_total_funding+10000000, 7,"Cloudera")

plt.axvline(Alibaba_total_funding,color='orange',linestyle ="--")
plt.text(Alibaba_total_funding+10000000, 4,"Alibaba")

plt.axvline(Verizon_total_funding,color='red',linestyle ="--")
plt.text(Verizon_total_funding+100000000, 15,"Verizon Communications")

plt.axvline(Sberbank_total_funding,color='mediumseagreen',linestyle ="--")
plt.text(Sberbank_total_funding+100000000, 12,"Sberbank")


plt.show()
```



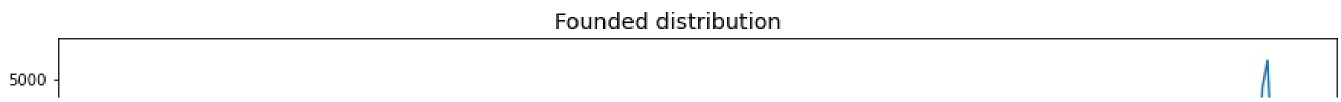
In [49]:
```
df['founded_at'].head()
```

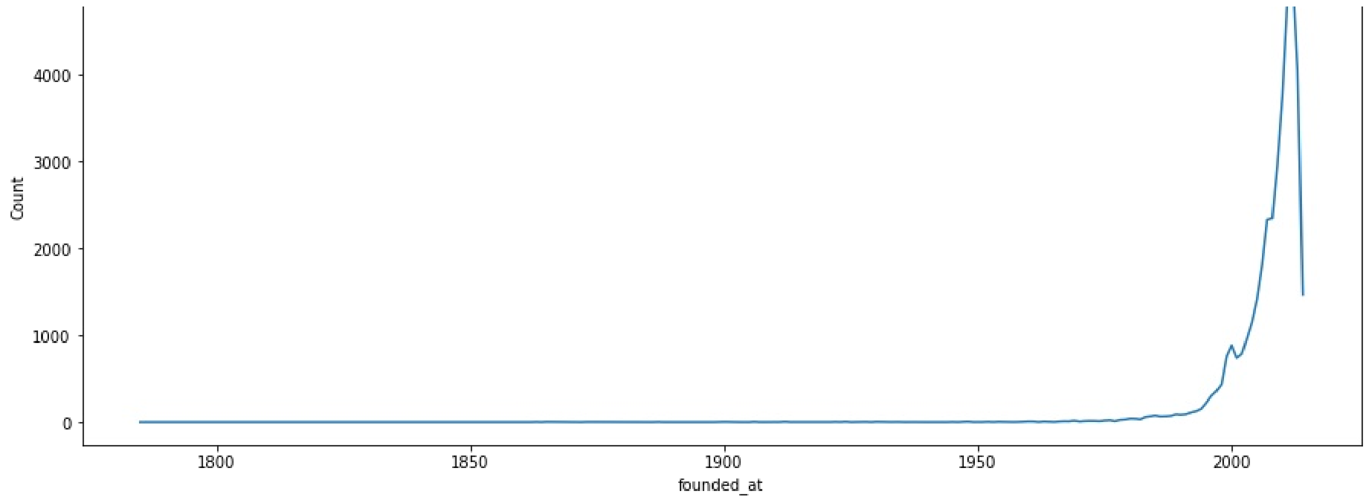
Out[49]:
```
0    2012-06-01
1           NaN
2    2012-10-26
3    2011-04-01
4    2014-01-01
Name: founded_at, dtype: object
```

This column is provided in term of string format which we need to convert to dateline first

In [51]:
```
df['founded_at'] = pd.to_datetime(df['founded_at'], errors = 'coerce')
```

In [52]:
```
plt.rcParams['figure.figsize'] = 15,6
df['name'].groupby(df["founded_at"].dt.year).count().plot(kind="line")

plt.ylabel('Count')
plt.title("Founded distribution ", fontdict=None, position= [0.48,1.05], size = 'x-large')
plt.show()
```



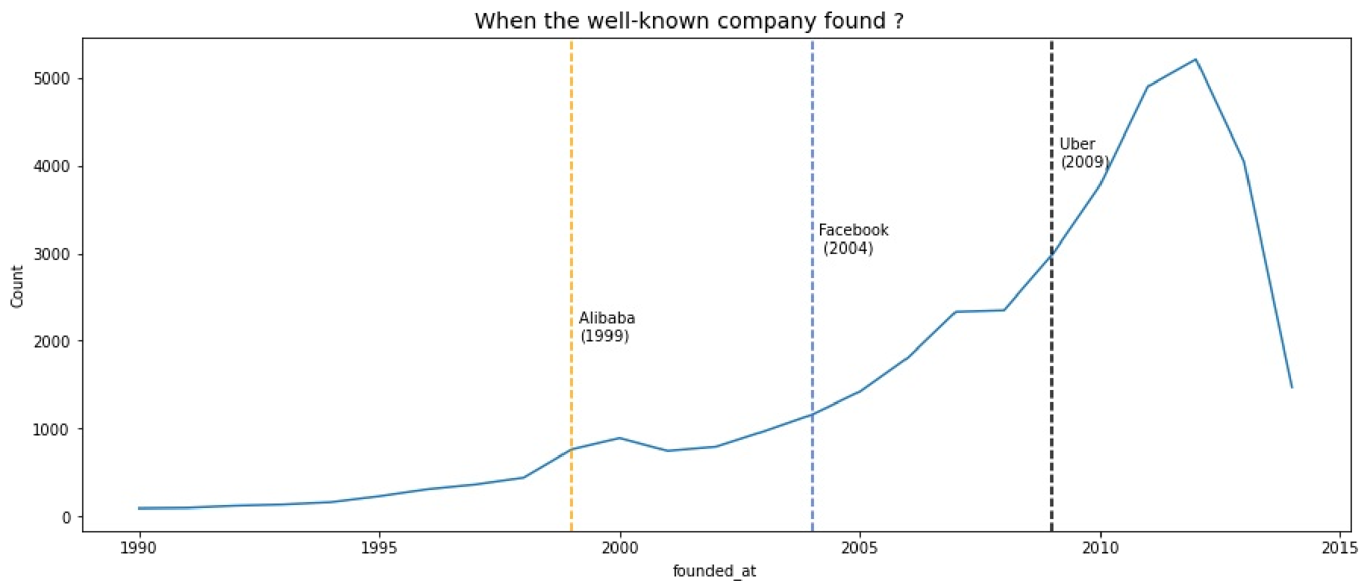


In [53]:
```
Facebook_founded_year = df['founded_at'][df['name']=="Facebook"].dt.year.values[0]
Uber_founded_year  = df['founded_at'][df['name']=="Uber"].dt.year.values[0]
Alibaba_founded_year  = df['founded_at'][df['name']=="Alibaba"].dt.year.values[0]
```

In [54]:
```
Uber_founded_year
```

Out[54]:
```
2009
```

In [56]:
```
plt.rcParams['figure.figsize'] = 15,6
df['name'][df["founded_at"].dt.year >= 1990].groupby(df["founded_at"].dt.year).count().plot(kind="line")
plt.ylabel('Count')

plt.axvline(Facebook_founded_year,color='royalblue',linestyle ="--")
plt.text(Facebook_founded_year+0.15, 3000,"Facebook \n (2004)")

plt.axvline(Uber_founded_year,color='black',linestyle ="--")
plt.text(Uber_founded_year+0.15, 4000,"Uber \n(2009)")

plt.axvline(Alibaba_founded_year,color='orange',linestyle ="--")
plt.text(Alibaba_founded_year+0.15, 2000,"Alibaba \n(1999)")


plt.title("When the well-known company found ?", fontdict=None, position= [0.48,1.05], size = 'x-large')
plt.show()
```



Country Code

In [57]:
```
len(df['country_code'].unique())
```

Out[57]:
```
116
```

In [58]:
```
df['country_code'].value_counts()[:20]
```

Out[58]:
```
USA    28793
GBR     2642
CAN     1405
CHN     1239
DEU      968
FRA      866
IND      849
ISR      682
ESP      549
RUS      368
SWE      315
AUS      314
ITA      308
NLD      307
IRL      306
SGP      299
BRA      293
CHL      285
JPN      284
KOR      246
Name: country_code, dtype: int64
```

In [59]:
```
df['count'] = 1
country_market = df[['count','country_code','market']].groupby(['country_code','market']).agg({'count': 'sum'})
# Change: groupby state_office and divide by sum
country_market_pct = country_market.groupby(level=0).apply(lambda x:
                                                 100 * x / float(x.sum()))
country_market_pct.reset_index(inplace = True)
```

In [62]:
```
USA_market_pct = country_market_pct[country_market_pct['country_code'] == "USA"]
USA_market_pct = USA_market_pct.sort_values('count',ascending = False)[0:10]
```

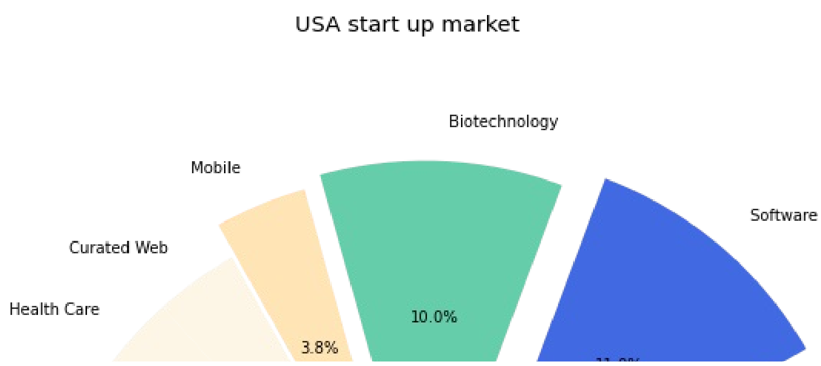
In [63]:
```
Ind_market_pct = country_market_pct[country_market_pct['country_code'] == "IND"]
Ind_market_pct['count'].sum()
```

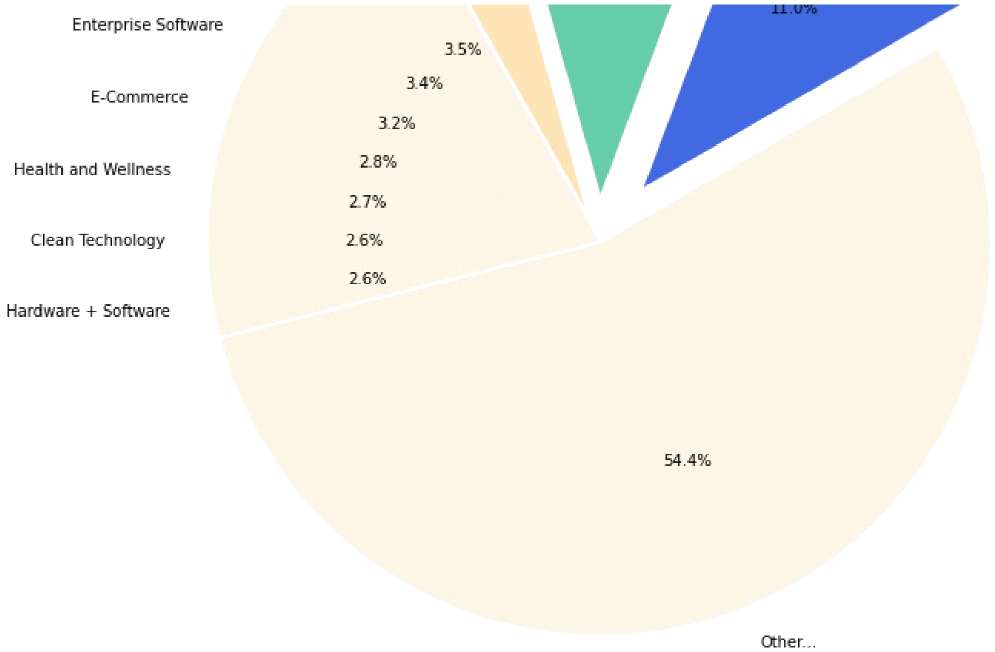
Out[63]:
```
100.0
```

In [64]:
```
USA_market_pct['count'].sum()
```

Out[64]:
```
45.5806726108994
```

In [65]:
```
## USA
plt.rcParams['figure.figsize'] =10,10
labels = list(USA_market_pct['market'])+['Other...']
sizes = list(USA_market_pct['count'])+[100-USA_market_pct['count'].sum()]
explode = (0.18, 0.12, 0.09,0,0,0,0,0,0,0,0.01)
colors =  ['royalblue','mediumaquamarine','moccasin'] +['oldlace']*8

plt.pie(sizes, explode = explode, colors = colors ,labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=30)
plt.axis('equal')
plt.tight_layout()
plt.title("USA start up market", fontdict=None, position= [0.48,1.1], size = 'x-large')

plt.show()
```

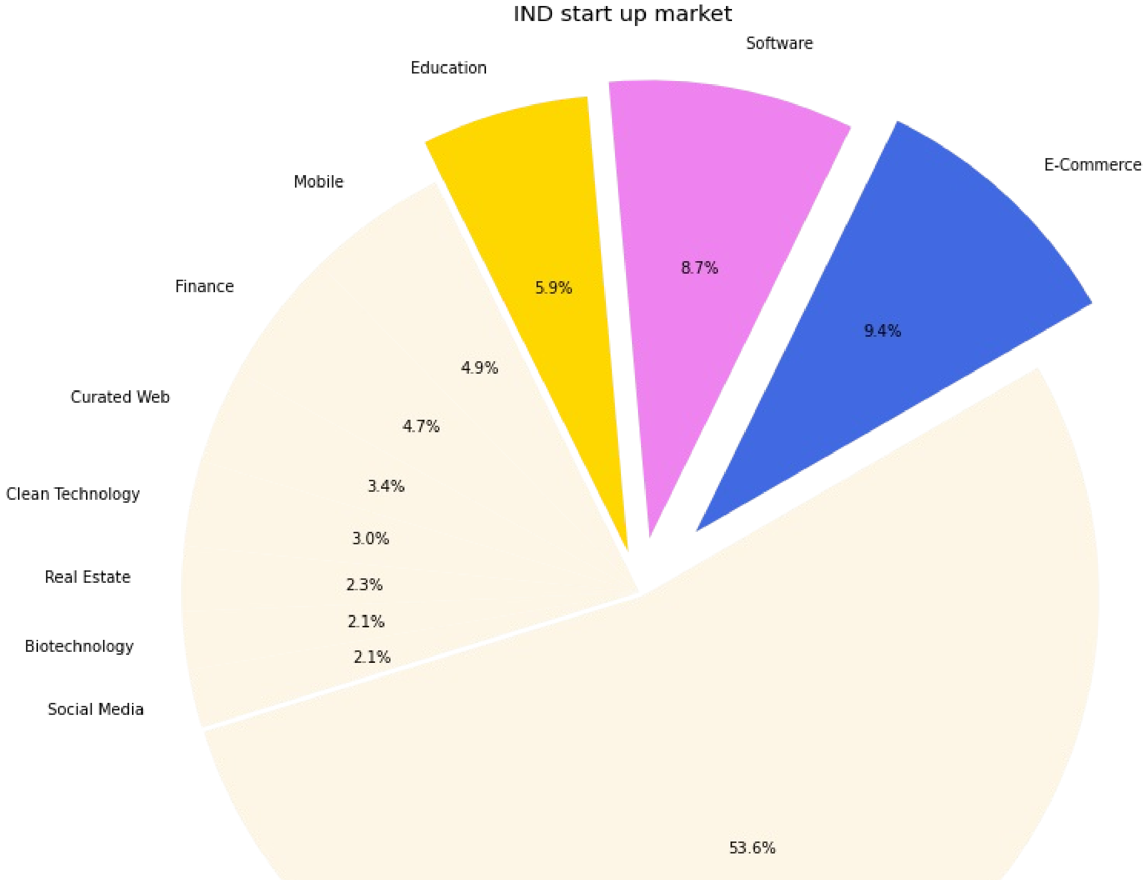




For USA, Most of start up market is about Software & Technology

In [66]:
```
IND_market_pct = country_market_pct[country_market_pct['country_code'] == "IND"]
IND_market_pct = IND_market_pct.sort_values('count',ascending = False)[0:10]
```

In [67]:
```
plt.rcParams['figure.figsize'] = 10,10
labels = list(IND_market_pct['market'])+['Other...']
sizes = list(IND_market_pct['count'])+[100-USA_market_pct['count'].sum()]
explode = (0.18, 0.12, 0.09,0,0,0,0,0,0,0,0.01)
colors =  ['royalblue','violet','gold'] +['oldlace']*8

plt.pie(sizes, explode = explode, colors = colors ,labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=30)
plt.axis('equal')
plt.tight_layout()
plt.title("IND start up market", fontdict=None, position= [0.48,1.1], size = 'x-large')
plt.show()
```



Funding Analysis

In [73]:

```
warnings.filterwarnings("ignore")
pd.options.display.float_format = '{:.2f}'.format
## Function for providing summary in dataframe
%matplotlib inline

def funding_information(df,name):
    company = df[df['name'] == name]
    print ("Company : ", name)
    print ("Total Funding : ", company.funding_total_usd.values[0] , " $")
    print ("Seed Funding : ", company.seed.values[0] , " $")
    print ("Angle Funding :", company.angel.values[0] , " $")
    print ("Grant Funding : ",company.grant.values[0] , " $")
    print ("Product Crowd Funding : ",company.product_crowdfunding.values[0] , " $")
    print ("Equity Crowd Funding : ",company.equity_crowdfunding.values[0] , " $")
    print ("Undisclode Funding : ", company.undisclosed.values[0] , " $")
    print ("Convertible Note : ", company.convertible_note.values[0] , " $")
    print ("Debt Financing : ", company.debt_financing.values[0] , " $")
    print ("Private Equity : ",company.private_equity.values[0] , " $")
    print ("PostIPO Equity : ",company.post_ipo_equity.values[0] , " $")
    print ("PostIPO Debt : ",company.post_ipo_debt.values[0] , " $")
    print ("Secondary Market : ",company.secondary_market.values[0] , " $")
    print ("Venture Funding : ",company.venture.values[0] , " $")
    print ("Round A funding : ",company.round_A.values[0] , " $")
    print ("Round B funding : ",company.round_B.values[0] , " $")
    print ("Round C funding : ",company.round_C.values[0] , " $")
    print ("Round D funding : ",company.round_D.values[0] , " $")
    print ("Round E funding : ",company.round_E.values[0] , " $")
    print ("Round F funding : ",company.round_F.values[0] , " $")
    print ("Round G funding : ",company.round_G.values[0] , " $")
    print ("Round H funding : ",company.round_H.values[0] , " $")

def count_word(df, ref_col, liste):
    keyword_count = dict()
    for s in liste: keyword_count[s] = 0
    for liste_keywords in df[ref_col].str.split('|'):
        if type(liste_keywords) == float and pd.isnull(liste_keywords): continue
        for s in [s for s in liste_keywords if s in liste]:
            if pd.notnull(s): keyword_count[s] += 1
    #______________________________________________________________________
    # convert the dictionary in a list to sort the keywords by frequency
    keyword_occurences = []
    for k,v in keyword_count.items():
        keyword_occurences.append([k,v])
    keyword_occurences.sort(key = lambda x:x[1], reverse = True)
    return keyword_occurences, keyword_count


def makeCloud(Dict,name,color):
    words = dict()

    for s in Dict:
        words[s[0]] = s[1]

        wordcloud = WordCloud(
                      width=1500,
                      height=750,
                      background_color=color,
                      max_words=50,
                      max_font_size=500,
                      normalize_plurals=False)
        wordcloud.generate_from_frequencies(words)


    fig = plt.figure(figsize=(12, 8))
    plt.title(name)
    plt.imshow(wordcloud)
    plt.axis('off')

    plt.show()
```

```
funding_information(df,"Dropbox")
```

```
Company :  Dropbox
Total Funding :  1107215000.0  $
Seed Funding :  15000.0  $
Angle Funding : 0.0  $
Grant Funding :  0.0  $
Product Crowd Funding :  0.0  $
Equity Crowd Funding :  0.0  $
Undisclode Funding :  0.0  $
Convertible Note :  0.0  $
Debt Financing :  500000000.0  $
Private Equity :  0.0  $
PostIPO Equity :  0.0  $
PostIPO Debt :  0.0  $
Secondary Market :  0.0  $
Venture Funding :  607200000.0  $
Round A funding :  7200000.0  $
Round B funding :  250000000.0  $
Round C funding :  350000000.0  $
Round D funding :  0.0  $
Round E funding :  0.0  $
Round F funding :  0.0  $
Round G funding :  0.0  $
Round H funding :  0.0  $
```

Here I have the print function to make data easier to consume, As you can see in "Dropbox" case, The total funding is 1,107,215,000 usd

which came from Seed Funding 15,000 usd and Debt Financing 500,000,000 usd and Venture Funding 6,0720,0000 usd

For Venture Funding, this dataset also shows How much company get fund in each round.

In [75]:
```
funding_information(df,"Uber")
```

```
Company :  Uber
Total Funding :  1507450000.0  $
Seed Funding :  200000.0  $
Angle Funding : 1250000.0  $
Grant Funding :  0.0  $
Product Crowd Funding :  0.0  $
Equity Crowd Funding :  0.0  $
Undisclode Funding :  0.0  $
Convertible Note :  0.0  $
Debt Financing :  0.0  $
Private Equity :  0.0  $
PostIPO Equity :  0.0  $
PostIPO Debt :  0.0  $
Secondary Market :  0.0  $
Venture Funding :  1506000000.0  $
Round A funding :  11000000.0  $
Round B funding :  37000000.0  $
Round C funding :  258000000.0  $
Round D funding :  1200000000.0  $
Round E funding :  0.0  $
Round F funding :  0.0  $
Round G funding :  0.0  $
Round H funding :  0.0  $
```

For the "Uber" case, The total funding is 1,507,450,000 usd which came from Seed Funding 200,000 usd and Angle Funding 1,250,000 usd and Venture Funding 1,506,000,000 usd Seed funding is the first official equity funding stage. It typically represents the first official money that a business venture or enterprise raises; some companies never extend beyond seed funding into Series A rounds or beyond.

In [76]:
```
df[['name','seed']].head(5)
```

Out[76]:

	name	seed
0	#waywire	1750000.00
1	&TV Communications	0.00
2	'Rock' Your Paper	40000.00
3	(In)Touch Network	1500000.00
4	-R- Ranch and Mine	0.00

Average funding in this stage ? Note you need to beware when use the mean value Most of value in this column is 0, they will drag your average value down
The solution is using data which is not 0 to find average

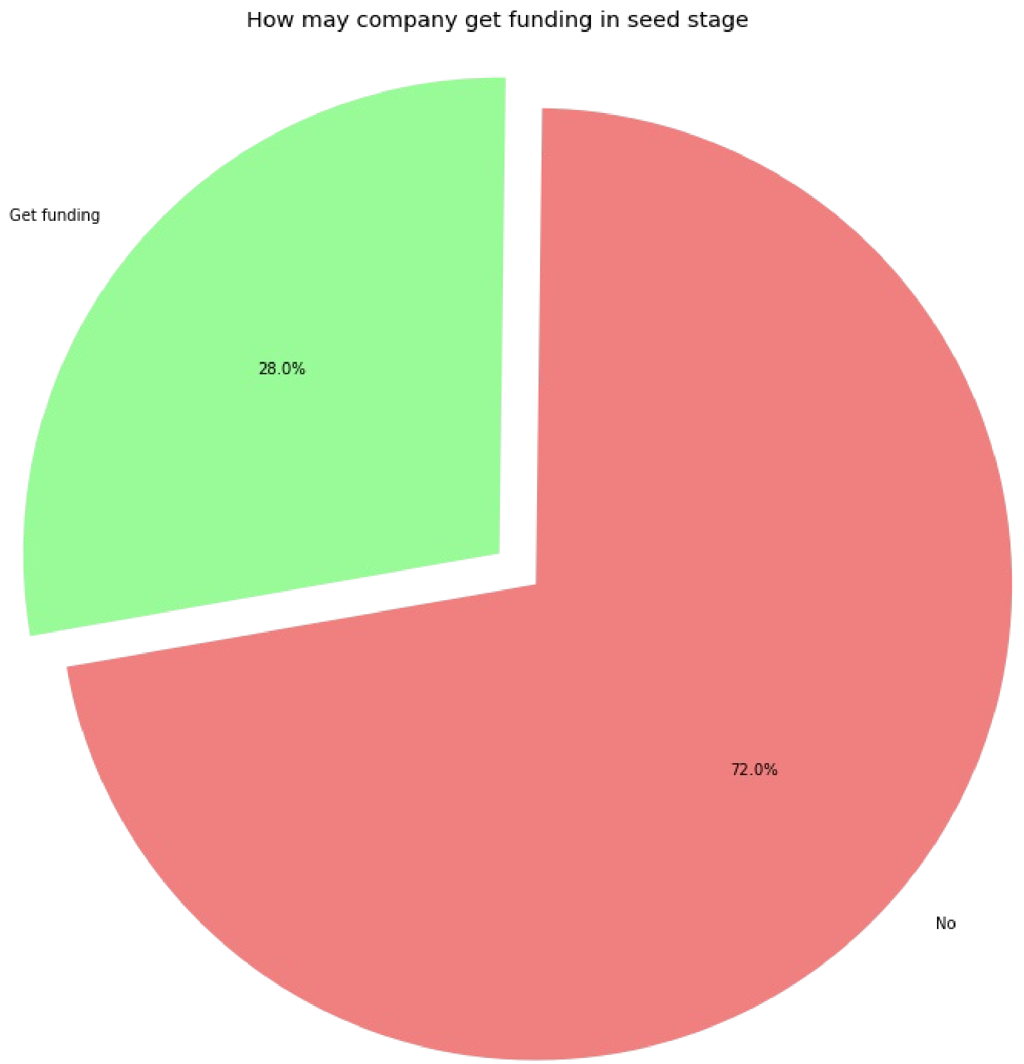
In [78]:
```
print("The average of seed funding stage is around ",df['seed'][df['seed'] != 0].mean(), "$")
```

```
The average of seed funding stage is around  776350.5418021533 $
```

But... How many company get funding in seed stage ?

In [79]:
```
df['get_funding_in_seed'] = df['seed'].map(lambda s :1  if s > 0 else 0)
```

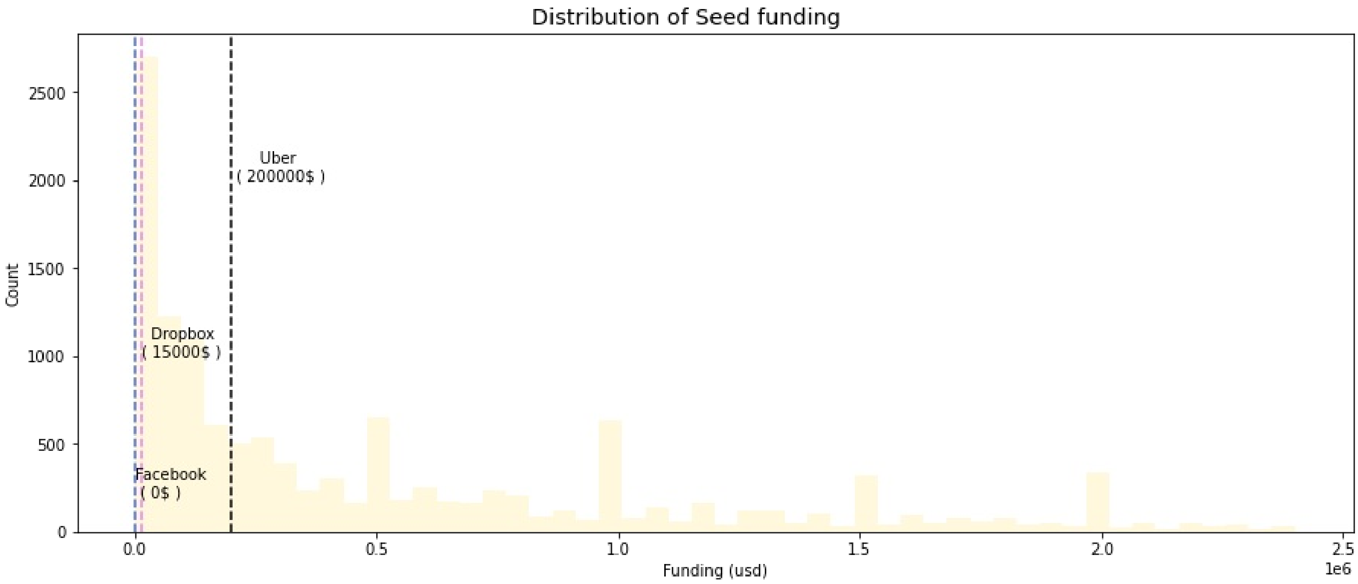
In [81]:
```
## USA
plt.rcParams['figure.figsize'] =10,10
labels = ['No','Get funding']
sizes = df['get_funding_in_seed'].value_counts().tolist()
explode = (0, 0.1)
colors =  ['lightcoral','palegreen']

plt.pie(sizes, explode = explode, colors = colors ,labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=190)
plt.axis('equal')
plt.tight_layout()
plt.title("How may company get funding in seed stage", fontdict=None, position= [0.48,1.1], size = 'x-large')

plt.show()
```



In [83]:
```
## Remove Outlier first
Q1 = df['seed'][df['seed'] != 0].quantile(0.25)
Q3 = df['seed'][df['seed'] != 0].quantile(0.75)
IQR = Q3 - Q1

lower_bound = (Q1 - 1.5 * IQR)
upper_bound = (Q3 + 1.5 * IQR)
without_outlier = df[(df['seed'] > lower_bound ) & (df['seed'] < upper_bound)]
```

In [84]:
```
Facebook_seed_funding = df['seed'][df['name']=="Facebook"].values[0]
Uber_seed_funding   = df['seed'][df['name']=="Uber"].values[0]
Dropbox_seed_funding   = df['seed'][df['name']=="Dropbox"].values[0]
```

In [85]:
```
plt.rcParams['figure.figsize'] = 15,6
plt.hist(without_outlier['seed'][without_outlier['seed']!=0].dropna(), bins=50,color = 'cornsilk' )

plt.axvline(Facebook_seed_funding,color='royalblue',linestyle ="--")
plt.text(Facebook_seed_funding+0.15, 200,"Facebook \n ( 0$ )")

plt.axvline(Uber_seed_funding,color='black',linestyle ="--")
plt.text(Uber_seed_funding+0.15, 2000,"      Uber \n ( 200000$ )")

plt.axvline(Dropbox_seed_funding,color='violet',linestyle ="--")
plt.text(Dropbox_seed_funding+0.15, 1000,"  Dropbox \n( 15000$ )")

plt.ylabel('Count')
plt.xlabel('Funding (usd)')
plt.title("Distribution of Seed funding ", fontdict=None, position= [0.48,1.05], size = 'x-large')
plt.show()
```



Angel funding

Who is angel?

An angel investor (also known as a private investor, seed investor or angel funder) is a high net worth individual who provides financial backing for small startups or entrepreneurs, typically in exchange for ownership equity in the company. Often, angel investors are found among an entrepreneur's family and friends. The funds that angel investors provide may be a one-time investment to help the business get off the ground or an ongoing injection to support and carry the company through its difficult early stages.

In [86]:
```
print("The average of Angel funding is around ",df['angel'][df['angel'] != 0].mean(), "$")
```

```
The average of Angel funding is around  1033615.6954298498 $
```

In [87]:
```
df['get_funding_in_angel'] = df['angel'].map(lambda s :"Get funding"  if s > 0 else "Not get funding")
```

In [88]:
```
print("Only " , df['get_funding_in_angel'].value_counts().values[1], " companies has angel investor")
print("while " , df['get_funding_in_angel'].value_counts().values[0], " are not")
print("~",df['get_funding_in_angel'].value_counts().values[1]/(df['get_funding_in_angel'].value_counts().values[1
```

```
Only  3129  companies has angel investor
while  46308  are not
~ 6.329267552642757 percent
```

Only 3129 companies has angel investor while 46308 are not ~ 6.329267552642757 percent Investment in each round

In [89]:
```
df['round_A'][df['round_A'] != 0].mean()
```

Out[89]: 6830906.178162835

In [90]:
```
df['round_B'][df['round_B'] != 0].mean()
```

Out[90]: 13549761.864145402

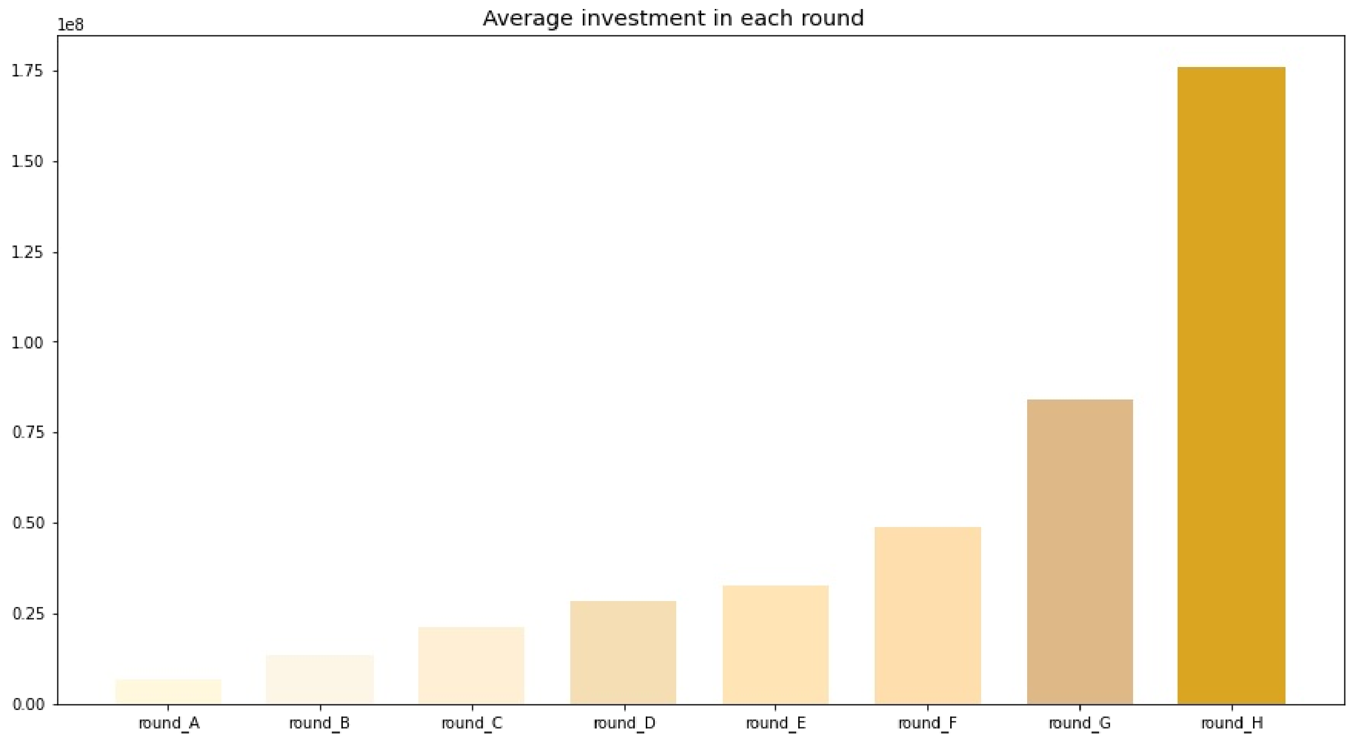
In [91]:
```
df['round_C'][df['round_C'] != 0].mean()
```

Out[91]: 21004716.314416636

Those number above is average funding of each round of investment, you can see that the farther round the investment is higher too.

In [92]:
```
round_ = ['round_A','round_B','round_C','round_D','round_E','round_F','round_G','round_H']
amount = [df['round_A'][df['round_A'] != 0].mean(),
          df['round_B'][df['round_B'] != 0].mean(),
          df['round_C'][df['round_C'] != 0].mean(),
          df['round_D'][df['round_D'] != 0].mean(),
          df['round_E'][df['round_E'] != 0].mean(),
          df['round_F'][df['round_F'] != 0].mean(),
          df['round_G'][df['round_G'] != 0].mean(),
         df['round_H'][df['round_H'] != 0].mean()]
```

In [93]:
```
plt.rcParams['figure.figsize'] = 15,8

height = amount
bars =  round_
y_pos = np.arange(len(bars))
plt.bar(y_pos, height , width=0.7, color= ['cornsilk','oldlace','papayawhip','wheat','moccasin','navajowhite','bu
plt.xticks(y_pos, bars)
plt.title("Average investment in each round", fontdict=None, position= [0.48,1.05], size = 'x-large')
plt.show()
```



In [94]:
```
# unique value in City
print("Number of Unique Values: ", (df['market'].nunique()))
print("Number of Missing Values: ", df['market'].isna().sum())
# Value Counts of top 10 cities
print((df['market'].value_counts()[0:10]))
```

```
Number of Unique Values:  753
Number of Missing Values:  3968
 Software                4620
 Biotechnology           3688
```

```
 Mobile                 1983
 E-Commerce             1805
 Curated Web            1655
 Enterprise Software    1280
 Health Care            1207
 Clean Technology       1200
 Games                  1182
 Hardware + Software    1081
Name: market, dtype: int64
```

In [95]:

```
df
```

Out[95]:

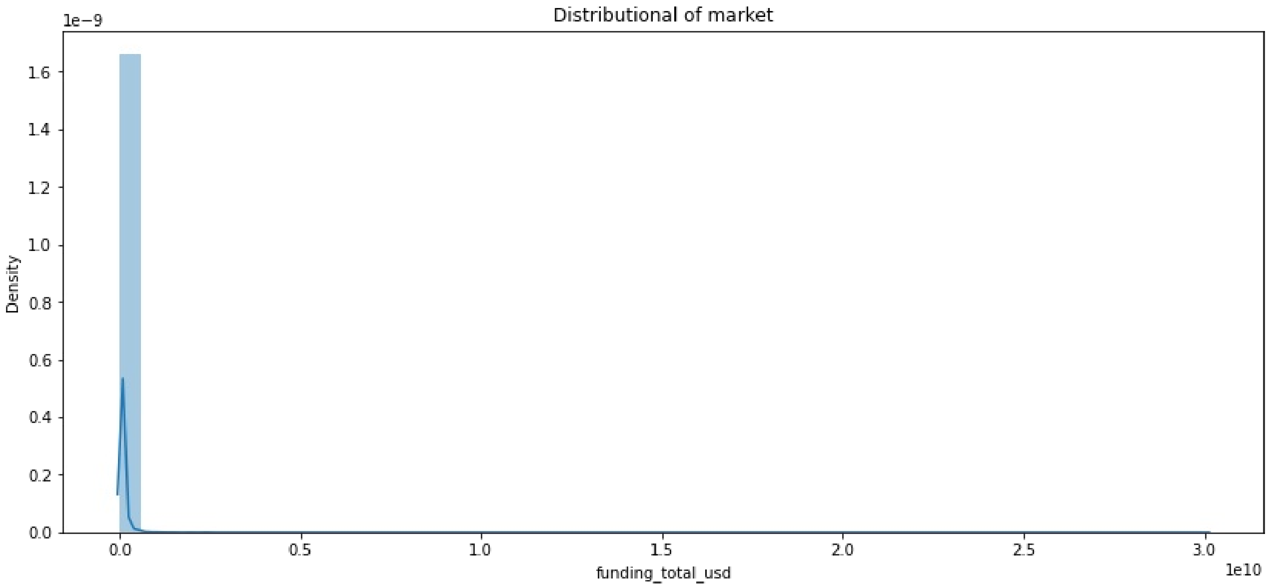
	permalink	name	homepage_url	category_list	market	funding_tota
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|News|	News	1750
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Games|	Games	4000
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Education|	Publishing	40
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Music|i...	Electronics	1500
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Games|	Tourism	60
...	...	...	...	...	...	
49433	/organization/zzish	Zzish	http://www.zzish.com	|Analytics|Gamification|Developer APIs|iOS|And...	Education	320
49434	/organization/zznode-science-and-technology-co...	ZZNode Science and Technology	http://www.zznode.com	|Enterprise Software|	Enterprise Software	1587
49435	/organization/zzzzapp-com	Zzzzapp Wireless ltd.	http://www.zzzzapp.com	|Web Development|Advertising|Wireless|Mobile|	Web Development	97
49436	/organization/a-list-games	[a]list games	http://www.alistgames.com	|Games|	Games	9300
49437	/organization/x	[x+1]	http://www.xplusone.com/	|Enterprise Software|	Enterprise Software	45000

49437 rows × 42 columns

In [96]:

```
#Plot of training_hours
print("Number of Missing Values: ", df['funding_total_usd'].isna().sum())
plt.figure(figsize=(14,6))
sns.distplot(df.funding_total_usd).set_title("Distributional of market");
```

```
Number of Missing Values:  8531
```



Feature Selection

Feature Selection

correlation in data

In [97]:
```
df.corr()
```

Out[97]:

	funding_total_usd	funding_rounds	founded_year	seed	venture	equity_crowdfunding	undisclosed	convertible_note	de
funding_total_usd	1.00	0.11	-0.07	-0.00	0.21	-0.00	0.02	0.01	
funding_rounds	0.11	1.00	-0.06	0.09	0.40	-0.00	0.03	0.02	
founded_year	-0.07	-0.06	1.00	0.08	-0.09	0.01	-0.04	-0.01	
seed	-0.00	0.09	0.08	1.00	-0.01	-0.00	-0.00	-0.00	
venture	0.21	0.40	-0.09	-0.01	1.00	-0.01	0.01	0.00	
equity_crowdfunding	-0.00	-0.00	0.01	-0.00	-0.01	1.00	-0.00	-0.00	
undisclosed	0.02	0.03	-0.04	-0.00	0.01	-0.00	1.00	-0.00	
convertible_note	0.01	0.02	-0.01	-0.00	0.00	-0.00	-0.00	1.00	
debt_financing	0.90	0.02	-0.03	-0.00	0.01	-0.00	-0.00	0.00	
angel	0.00	0.06	0.02	-0.00	0.01	0.02	0.00	-0.00	
grant	0.04	0.01	-0.09	-0.01	0.01	-0.00	-0.00	-0.00	
private_equity	0.23	0.06	-0.06	-0.01	0.06	-0.00	0.01	0.01	
post_ipo_equity	0.23	0.02	-0.04	-0.00	0.01	-0.00	0.00	0.00	
post_ipo_debt	0.26	-0.00	-0.03	-0.00	-0.00	-0.00	-0.00	0.00	
secondary_market	0.04	0.01	-0.01	-0.00	0.06	-0.00	-0.00	-0.00	
product_crowdfunding	0.00	0.02	-0.00	0.20	-0.00	0.01	-0.00	-0.00	
round_A	0.06	0.17	-0.02	0.01	0.33	-0.00	0.00	-0.00	
round_B	0.10	0.28	-0.04	0.00	0.50	-0.01	-0.00	0.00	
round_C	0.13	0.30	-0.05	-0.00	0.58	-0.00	0.00	0.00	
round_D	0.12	0.20	-0.03	-0.01	0.59	-0.00	0.00	0.00	
round_E	0.11	0.20	-0.03	-0.01	0.53	-0.00	0.03	0.00	
round_F	0.09	0.10	-0.01	-0.01	0.43	-0.00	-0.00	-0.00	
round_G	0.08	0.06	-0.00	-0.00	0.42	-0.00	-0.00	-0.00	
round_H	0.07	0.04	-0.00	-0.00	0.37	-0.00	-0.00	-0.00	
count	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
get_funding_in_seed	-0.05	0.03	0.27	0.33	-0.12	-0.01	-0.02	-0.01	

26 rows × 26 columns

In [98]:
```
df.head()
```

Out[98]:

	permalink	name	homepage_url	category_list	market	funding_total_usd
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|News|	News	1750000.00
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Games|	Games	4000000.00
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Education|	Publishing	40000.00
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Music|i...	Electronics	1500000.00
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Games|	Tourism	60000.00

5 rows × 42 columns

Splitting the data

In [99]:
```
# Non_numerical_data
df.homepage_url
```

```
df.category_list
df.market
df.status
df.name
df.country_code
df.state_code
df.region
df.city
df.founded_at
```

Out[99]:

```
0       2012-06-01
1              NaT
2       2012-10-26
3       2011-04-01
4       2014-01-01
           ...
49433   2013-01-28
49434          NaT
49435   2012-05-13
49436          NaT
49437   1999-01-01
Name: founded_at, Length: 49437, dtype: datetime64[ns]
```

In [100…

```
# Numerical Data
df.funding_total_usd
df.funding_rounds
df.first_funding_at
df.last_funding_at
df.seed
df.venture
df.equity_crowdfunding
df.undisclosed
df.convertible_note
df.debt_financing
df.angel
df.grant
df.private_equity
df.post_ipo_debt
df.secondary_market
df.product_crowdfunding
df.round_A
df.round_B
df.round_C
df.round_D
df.round_E
df.round_F
df.round_G
df.round_H
df.grant
```

Out[100…

```
0       0.00
1       0.00
2       0.00
3       0.00
4       0.00
        ...
49433   0.00
49434   0.00
49435   0.00
49436   0.00
49437   0.00
Name: grant, Length: 49437, dtype: float64
```

In [101…

```
# 'funding_total_usd',
# 'name','homepage_url','category_list','market','status', 'country_code', 'state_code' ,'region','city''founded_

df.grant
```

Out[101…

```
0       0.00
1       0.00
2       0.00
3       0.00
4       0.00
        ...
49433   0.00
49434   0.00
49435   0.00
49436   0.00
49437   0.00
Name: grant, Length: 49437, dtype: float64
```

In [102…

```
X=df[['funding_rounds', 'seed' ,'venture','equity_crowdfunding','undisclosed' ,'convertible_note' ,'debt_financir
y=df['grant']
```

In [103…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [104…

```
X_train.fillna(0)
X.fillna(0)
X_train.isnull().sum()
```

Out[104…

```
funding_rounds         0
seed                   0
venture                0
equity_crowdfunding    0
undisclosed            0
convertible_note       0
debt_financing         0
angel                  0
grant                  0
private_equity         0
post_ipo_equity        0
post_ipo_debt          0
secondary_market       0
product_crowdfunding   0
round_A                0
round_B                0
round_C                0
round_D                0
round_E                0
round_F                0
round_G                0
round_H                0
dtype: int64
```

Applying Models

Determine the baseline model accuracy

In [105…

```
# create baseline model
def baseline_model(n_predictions, value_to_predict):
    """
    just predict a single value (e.g. mean) for everything
    """
    baseline_preds = []
    for i in range(n_predictions):
        baseline_preds.append(value_to_predict)
    return pd.Series(baseline_preds)
```

In [106…

```
n_predictions = len(y_test)  # how many predictions to make? '3832'
baseline_value =pd.Series(y_train).value_counts().index[0]  # what value to predict? (classification = most comm
baseline_preds = baseline_model(n_predictions, baseline_value)
baseline_preds = baseline_model(n_predictions, baseline_value)
baseline_preds
```

Out[106…

```
0       0.00
1       0.00
2       0.00
3       0.00
4       0.00
        ...
9883    0.00
9884    0.00
9885    0.00
9886    0.00
9887    0.00
Length: 9888, dtype: float64
```

In [107…

```
n_predictions = len(y_test)  # how many predictions to make? '3832'
baseline_value =pd.Series(y_train).value_counts().index[0]  # what value to predict? (classification = most comm
baseline_preds = baseline_model(n_predictions, baseline_value)
baseline_preds = baseline_model(n_predictions, baseline_value)
baseline_preds
```

Out[107…

```
0       0.00
```

```
1       0.00
2       0.00
3       0.00
4       0.00
        ...
9883    0.00
9884    0.00
9885    0.00
9886    0.00
9887    0.00
Length: 9888, dtype: float64
```

In [108…

```
baseline_acc=accuracy_score(y_test, baseline_preds) #Accuracy score of baseline model
print('The baseline model accuracy score is :',baseline_acc)
```

```
The baseline model accuracy score is : 0.9765372168284789
```

RandomForestClassifier Model

In [109…

```
# create and fit RandomForestClassifier model
rfc=RandomForestClassifier()
rfc.fit(X_train, y_train)
# predict
pred = rfc.predict(X_test)
# pred
rfc_acc= accuracy_score(y_test, pred)
print('The accuracy score using the RandomForestClassifier (befor resample) is :',rfc_acc)
print(classification_report(y_test, pred))
```

```
The accuracy score using the RandomForestClassifier (befor resample) is : 0.9851334951456311
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      9656
      1003.0       0.00      0.00      0.00         0
      1700.0       0.00      0.00      0.00         1
      1840.0       0.00      0.00      0.00         0
      2000.0       0.00      0.00      0.00         2
      2393.0       0.00      0.00      0.00         0
      2500.0       0.00      0.00      0.00         1
      9214.0       0.00      0.00      0.00         0
     10000.0       1.00      1.00      1.00         2
     10009.0       0.00      0.00      0.00         1
     12000.0       0.00      0.00      0.00         0
     12500.0       0.00      0.00      0.00         0
     13000.0       0.00      0.00      0.00         2
     14852.0       0.00      0.00      0.00         1
     15000.0       1.00      1.00      1.00         1
     17215.0       0.00      0.00      0.00         0
     18000.0       0.00      0.00      0.00         1
     18394.0       0.00      0.00      0.00         1
     20000.0       1.00      1.00      1.00         1
     25000.0       0.83      0.71      0.77         7
     26500.0       0.00      0.00      0.00         1
     29000.0       0.00      0.00      0.00         1
     30000.0       0.00      0.00      0.00         2
     31777.0       0.00      0.00      0.00         1
     32521.0       0.00      0.00      0.00         0
     32707.0       0.00      0.00      0.00         1
     33000.0       0.00      0.00      0.00         0
     34060.0       1.00      1.00      1.00         1
     35000.0       1.00      1.00      1.00         2
     37532.0       0.00      0.00      0.00         0
     39700.0       0.00      0.00      0.00         1
     40000.0       0.83      0.83      0.83         6
     42000.0       0.00      0.00      0.00         0
     43000.0       0.00      0.00      0.00         0
     43823.0       1.00      1.00      1.00         1
     45000.0       0.00      0.00      0.00         1
     45412.0       0.00      0.00      0.00         1
     45457.0       0.00      0.00      0.00         1
     45685.0       0.00      0.00      0.00         0
     47500.0       0.00      0.00      0.00         1
     48387.0       0.00      0.00      0.00         0
     49000.0       0.00      0.00      0.00         0
     50000.0       0.71      1.00      0.83         5
     50408.0       0.00      0.00      0.00         1
     53356.0       0.00      0.00      0.00         0
     55407.0       0.00      0.00      0.00         0
     60000.0       1.00      1.00      1.00         1
```

62000.0	0.00	0.00	0.00	0
67000.0	0.00	0.00	0.00	1
75000.0	0.00	0.00	0.00	3
86000.0	0.00	0.00	0.00	0
86729.0	0.00	0.00	0.00	1
90000.0	0.00	0.00	0.00	1
90019.0	0.00	0.00	0.00	1
95000.0	0.00	0.00	0.00	1
96325.0	0.00	0.00	0.00	1
96832.0	0.00	0.00	0.00	0
97500.0	0.00	0.00	0.00	0
100000.0	0.83	0.83	0.83	6
106536.0	0.00	0.00	0.00	0
115000.0	0.00	0.00	0.00	1
120000.0	0.00	0.00	0.00	0
120941.0	0.00	0.00	0.00	1
129390.0	0.00	0.00	0.00	0
130295.0	0.00	0.00	0.00	1
136700.0	0.00	0.00	0.00	1
145000.0	0.00	0.00	0.00	0
145774.0	0.00	0.00	0.00	1
150000.0	0.67	1.00	0.80	2
154320.0	0.00	0.00	0.00	1
155000.0	0.00	0.00	0.00	0
159000.0	0.00	0.00	0.00	0
162000.0	0.00	0.00	0.00	1
163000.0	0.00	0.00	0.00	0
170000.0	0.00	0.00	0.00	1
180000.0	1.00	0.50	0.67	2
185005.0	0.00	0.00	0.00	0
190000.0	0.00	0.00	0.00	1
196000.0	0.00	0.00	0.00	1
197634.0	0.00	0.00	0.00	1
200000.0	0.50	0.50	0.50	2
203171.0	0.00	0.00	0.00	1
205000.0	0.00	0.00	0.00	1
213000.0	0.00	0.00	0.00	1
213094.0	0.00	0.00	0.00	0
214250.0	0.00	0.00	0.00	1
220099.0	0.00	0.00	0.00	1
225000.0	0.00	0.00	0.00	0
225195.0	0.00	0.00	0.00	1
230000.0	0.00	0.00	0.00	0
240000.0	0.00	0.00	0.00	1
244479.0	0.00	0.00	0.00	0
245000.0	0.00	0.00	0.00	1
246000.0	0.00	0.00	0.00	1
250000.0	0.30	1.00	0.46	3
253000.0	0.00	0.00	0.00	1
258043.0	0.00	0.00	0.00	1
262000.0	0.00	0.00	0.00	0
264000.0	0.00	0.00	0.00	0
285000.0	0.00	0.00	0.00	1
289256.0	0.00	0.00	0.00	0
293114.0	0.00	0.00	0.00	1
300000.0	0.00	0.00	0.00	2
349667.0	0.00	0.00	0.00	1
350000.0	0.00	0.00	0.00	2
352957.0	0.00	0.00	0.00	1
375000.0	1.00	1.00	1.00	2
378812.0	0.00	0.00	0.00	1
380000.0	0.00	0.00	0.00	0
382000.0	0.00	0.00	0.00	0
383000.0	0.00	0.00	0.00	0
400000.0	0.00	0.00	0.00	0
410172.0	0.00	0.00	0.00	0
428000.0	0.00	0.00	0.00	1
450000.0	0.00	0.00	0.00	1
480000.0	0.00	0.00	0.00	0
500000.0	0.50	0.80	0.62	5
522923.0	0.00	0.00	0.00	1
530000.0	0.00	0.00	0.00	1
537156.0	0.00	0.00	0.00	0
540000.0	0.00	0.00	0.00	1
545000.0	0.00	0.00	0.00	1
600000.0	1.00	1.00	1.00	1
620000.0	0.00	0.00	0.00	1
626995.0	0.00	0.00	0.00	0
638000.0	0.00	0.00	0.00	1
685000.0	0.00	0.00	0.00	0
700000.0	1.00	1.00	1.00	1
706000.0	0.00	0.00	0.00	1
721481.0	0.00	0.00	0.00	0
726694.0	0.00	0.00	0.00	1
729272.0	0.00	0.00	0.00	0
745000.0	0.00	0.00	0.00	1
750000.0	0.67	0.67	0.67	3
796180.0	0.00	0.00	0.00	1
799430.0	0.00	0.00	0.00	1

800000.0	0.00	0.00	0.00	1
850000.0	0.00	0.00	0.00	0
878000.0	0.00	0.00	0.00	1
903915.0	0.00	0.00	0.00	1
950000.0	0.00	0.00	0.00	0
975000.0	0.00	0.00	0.00	1
977000.0	0.00	0.00	0.00	1
984913.0	0.00	0.00	0.00	0
1000000.0	0.36	1.00	0.53	5
1100000.0	0.00	0.00	0.00	0
1150000.0	0.00	0.00	0.00	1
1158093.0	0.00	0.00	0.00	1
1200000.0	0.67	0.40	0.50	5
1289342.0	0.00	0.00	0.00	0
1370000.0	0.00	0.00	0.00	0
1400000.0	0.33	0.50	0.40	2
1475614.0	0.00	0.00	0.00	0
1500000.0	0.50	0.40	0.44	5
1507200.0	0.00	0.00	0.00	0
1600000.0	0.50	1.00	0.67	1
1690000.0	0.00	0.00	0.00	1
1700000.0	0.00	0.00	0.00	0
1725000.0	0.00	0.00	0.00	1
2000000.0	1.00	0.30	0.46	10
2100000.0	1.00	1.00	1.00	1
2108120.0	0.00	0.00	0.00	0
2200000.0	1.00	0.50	0.67	2
2300000.0	1.00	0.50	0.67	2
2440000.0	0.00	0.00	0.00	1
2500000.0	1.00	1.00	1.00	3
2547000.0	0.00	0.00	0.00	1
2600000.0	0.00	0.00	0.00	1
2800000.0	0.00	0.00	0.00	1
2814100.0	0.00	0.00	0.00	1
3000000.0	0.67	0.67	0.67	6
3500000.0	1.00	1.00	1.00	1
3663934.0	0.00	0.00	0.00	1
3684289.0	0.00	0.00	0.00	1
3859800.0	0.00	0.00	0.00	0
3900000.0	0.00	0.00	0.00	2
4362844.0	0.00	0.00	0.00	0
4450000.0	0.00	0.00	0.00	0
4500000.0	0.00	0.00	0.00	1
4800000.0	0.00	0.00	0.00	1
4900000.0	0.00	0.00	0.00	1
5000000.0	1.00	1.00	1.00	3
5303380.0	0.00	0.00	0.00	0
5789700.0	0.00	0.00	0.00	0
6000000.0	0.60	0.75	0.67	4
6200000.0	0.50	0.50	0.50	2
6300000.0	0.00	0.00	0.00	1
6527820.0	0.00	0.00	0.00	1
6900000.0	1.00	1.00	1.00	1
7000000.0	0.00	0.00	0.00	0
7090000.0	0.00	0.00	0.00	1
7500000.0	0.00	0.00	0.00	1
8900000.0	0.00	0.00	0.00	1
8923600.0	0.00	0.00	0.00	0
9000000.0	0.50	1.00	0.67	1
9100000.0	0.00	0.00	0.00	1
9150000.0	0.00	0.00	0.00	1
9450000.0	0.00	0.00	0.00	0
9500000.0	0.00	0.00	0.00	2
10000000.0	1.00	1.00	1.00	1
11000000.0	0.50	1.00	0.67	1
11100000.0	0.00	0.00	0.00	1
11200000.0	0.00	0.00	0.00	1
11400000.0	0.00	0.00	0.00	1
11750000.0	0.00	0.00	0.00	0
13600000.0	0.00	0.00	0.00	0
13836170.0	0.00	0.00	0.00	1
14000000.0	0.00	0.00	0.00	0
15000000.0	0.00	0.00	0.00	1
15152514.0	0.00	0.00	0.00	1
15300000.0	0.00	0.00	0.00	0
17100000.0	0.00	0.00	0.00	0
24600000.0	0.00	0.00	0.00	1
25000000.0	1.00	1.00	1.00	1
26000000.0	0.00	0.00	0.00	1
27400000.0	0.00	0.00	0.00	1
40000000.0	0.00	0.00	0.00	0
45000000.0	0.00	0.00	0.00	1
45100000.0	0.00	0.00	0.00	0
50000000.0	0.00	0.00	0.00	2
53500000.0	0.00	0.00	0.00	0
71600000.0	0.00	0.00	0.00	0
90000000.0	1.00	1.00	1.00	2
154053900.0	0.00	0.00	0.00	0
170000000.0	0.00	0.00	0.00	1

```
 191000000.0       0.00      0.00      0.00         1
 206000000.0       0.00      0.00      0.00         0

    accuracy                           0.99      9888
   macro avg       0.15      0.16      0.15      9888
weighted avg       0.98      0.99      0.98      9888
```

DecisionTreeClassifier Model

In [110…

```python
# create and fit DecisionTreeClassifier model
dtc=DecisionTreeClassifier()
dtc.fit(X_train,y_train)
#predict
pred = dtc.predict(X_test)
# pred

dtc_acc= accuracy_score(y_test, pred)
print('The accuracy score with using the decision tree classifier is :',dtc_acc)

print(classification_report(y_test, pred))
```

```
The accuracy score with using the decision tree classifier is : 0.9896844660194175
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      9656
      1003.0       0.00      0.00      0.00         0
      1700.0       0.00      0.00      0.00         1
      1840.0       0.00      0.00      0.00         0
      2000.0       0.00      0.00      0.00         2
      2393.0       0.00      0.00      0.00         0
      2500.0       0.00      0.00      0.00         1
     10000.0       0.67      1.00      0.80         2
     10009.0       0.00      0.00      0.00         1
     12500.0       0.00      0.00      0.00         0
     13000.0       0.00      0.00      0.00         2
     13300.0       0.00      0.00      0.00         0
     14852.0       0.00      0.00      0.00         1
     15000.0       0.50      1.00      0.67         1
     17215.0       0.00      0.00      0.00         0
     18000.0       0.00      0.00      0.00         1
     18394.0       0.00      0.00      0.00         1
     19175.0       0.00      0.00      0.00         0
     20000.0       1.00      1.00      1.00         1
     25000.0       1.00      1.00      1.00         7
     26000.0       0.00      0.00      0.00         0
     26500.0       0.00      0.00      0.00         1
     29000.0       0.00      0.00      0.00         1
     30000.0       0.67      1.00      0.80         2
     31777.0       0.00      0.00      0.00         1
     32094.0       0.00      0.00      0.00         0
     32521.0       0.00      0.00      0.00         0
     32707.0       0.00      0.00      0.00         1
     34060.0       1.00      1.00      1.00         1
     35000.0       1.00      1.00      1.00         2
     39700.0       0.00      0.00      0.00         1
     40000.0       0.86      1.00      0.92         6
     43823.0       1.00      1.00      1.00         1
     45000.0       1.00      1.00      1.00         1
     45412.0       0.00      0.00      0.00         1
     45457.0       0.00      0.00      0.00         1
     45685.0       0.00      0.00      0.00         0
     47500.0       0.00      0.00      0.00         1
     49000.0       0.00      0.00      0.00         0
     50000.0       1.00      1.00      1.00         5
     50408.0       0.00      0.00      0.00         1
     50576.0       0.00      0.00      0.00         0
     60000.0       1.00      1.00      1.00         1
     67000.0       1.00      1.00      1.00         1
     75000.0       1.00      1.00      1.00         3
     86729.0       0.00      0.00      0.00         1
     87206.0       0.00      0.00      0.00         0
     90000.0       0.50      1.00      0.67         1
     90019.0       0.00      0.00      0.00         1
     95000.0       0.00      0.00      0.00         1
     96325.0       0.00      0.00      0.00         1
     96832.0       0.00      0.00      0.00         0
     97500.0       0.00      0.00      0.00         0
    100000.0       1.00      1.00      1.00         6
    115000.0       1.00      1.00      1.00         1
    120000.0       0.00      0.00      0.00         0
    120941.0       0.00      0.00      0.00         1
    129390.0       0.00      0.00      0.00         0
```

130295.0	0.00	0.00	0.00	1
136700.0	0.00	0.00	0.00	1
143363.0	0.00	0.00	0.00	0
145000.0	0.00	0.00	0.00	0
145774.0	0.00	0.00	0.00	1
150000.0	1.00	1.00	1.00	2
154320.0	0.00	0.00	0.00	1
155000.0	0.00	0.00	0.00	0
162000.0	0.00	0.00	0.00	1
163000.0	0.00	0.00	0.00	0
170000.0	0.00	0.00	0.00	1
175200.0	0.00	0.00	0.00	0
180000.0	1.00	0.50	0.67	2
185005.0	0.00	0.00	0.00	0
190000.0	0.00	0.00	0.00	1
193888.0	0.00	0.00	0.00	0
196000.0	0.00	0.00	0.00	1
197634.0	0.00	0.00	0.00	1
200000.0	0.50	1.00	0.67	2
203171.0	0.00	0.00	0.00	1
205000.0	0.00	0.00	0.00	1
209800.0	0.00	0.00	0.00	0
213000.0	0.00	0.00	0.00	1
213094.0	0.00	0.00	0.00	0
214250.0	0.00	0.00	0.00	1
220099.0	0.00	0.00	0.00	1
225000.0	0.00	0.00	0.00	0
225195.0	0.00	0.00	0.00	1
229900.0	0.00	0.00	0.00	0
240000.0	0.00	0.00	0.00	1
244500.0	0.00	0.00	0.00	0
245000.0	0.00	0.00	0.00	1
246000.0	0.00	0.00	0.00	1
250000.0	0.75	1.00	0.86	3
253000.0	0.00	0.00	0.00	1
258043.0	0.00	0.00	0.00	1
264000.0	0.00	0.00	0.00	0
264933.0	0.00	0.00	0.00	0
285000.0	0.00	0.00	0.00	1
289256.0	0.00	0.00	0.00	0
293114.0	0.00	0.00	0.00	1
300000.0	1.00	1.00	1.00	2
349667.0	0.00	0.00	0.00	1
350000.0	0.50	1.00	0.67	2
352957.0	0.00	0.00	0.00	1
361165.0	0.00	0.00	0.00	0
375000.0	1.00	1.00	1.00	2
378812.0	0.00	0.00	0.00	1
410172.0	0.00	0.00	0.00	0
428000.0	0.00	0.00	0.00	1
450000.0	0.00	0.00	0.00	1
480000.0	0.00	0.00	0.00	0
500000.0	1.00	1.00	1.00	5
522923.0	0.00	0.00	0.00	1
530000.0	0.00	0.00	0.00	1
537156.0	0.00	0.00	0.00	0
537980.0	0.00	0.00	0.00	0
540000.0	0.00	0.00	0.00	1
545000.0	0.00	0.00	0.00	1
600000.0	1.00	1.00	1.00	1
620000.0	0.00	0.00	0.00	1
638000.0	0.00	0.00	0.00	1
663195.0	0.00	0.00	0.00	0
700000.0	1.00	1.00	1.00	1
706000.0	0.00	0.00	0.00	1
721481.0	0.00	0.00	0.00	0
726694.0	0.00	0.00	0.00	1
745000.0	0.00	0.00	0.00	1
750000.0	1.00	1.00	1.00	3
796180.0	0.00	0.00	0.00	1
799430.0	0.00	0.00	0.00	1
800000.0	0.33	1.00	0.50	1
878000.0	0.00	0.00	0.00	1
893883.0	0.00	0.00	0.00	0
903915.0	0.00	0.00	0.00	1
905000.0	0.00	0.00	0.00	0
975000.0	0.00	0.00	0.00	1
977000.0	0.00	0.00	0.00	1
984913.0	0.00	0.00	0.00	0
1000000.0	1.00	1.00	1.00	5
1100000.0	0.00	0.00	0.00	0
1140000.0	0.00	0.00	0.00	0
1150000.0	0.00	0.00	0.00	1
1158093.0	0.00	0.00	0.00	1
1200000.0	1.00	1.00	1.00	5
1400000.0	1.00	1.00	1.00	2
1500000.0	1.00	1.00	1.00	5
1600000.0	1.00	1.00	1.00	1
1680000.0	0.00	0.00	0.00	0

```
   1690000.0       0.00      0.00      0.00         1
   1700000.0       0.00      0.00      0.00         0
   1725000.0       0.00      0.00      0.00         1
   2000000.0       1.00      1.00      1.00        10
   2100000.0       1.00      1.00      1.00         1
   2108120.0       0.00      0.00      0.00         0
   2200000.0       1.00      0.50      0.67         2
   2300000.0       1.00      0.50      0.67         2
   2314688.0       0.00      0.00      0.00         0
   2400000.0       0.00      0.00      0.00         0
   2440000.0       0.00      0.00      0.00         1
   2500000.0       0.75      1.00      0.86         3
   2547000.0       0.00      0.00      0.00         1
   2600000.0       1.00      1.00      1.00         1
   2800000.0       0.50      1.00      0.67         1
   2814100.0       0.00      0.00      0.00         1
   3000000.0       1.00      1.00      1.00         6
   3500000.0       1.00      1.00      1.00         1
   3663934.0       0.00      0.00      0.00         1
   3671000.0       0.00      0.00      0.00         0
   3684289.0       0.00      0.00      0.00         1
   3859800.0       0.00      0.00      0.00         0
   3900000.0       0.00      0.00      0.00         2
   4450000.0       0.00      0.00      0.00         0
   4500000.0       0.00      0.00      0.00         1
   4800000.0       1.00      1.00      1.00         1
   4900000.0       1.00      1.00      1.00         1
   5000000.0       1.00      1.00      1.00         3
   6000000.0       1.00      1.00      1.00         4
   6200000.0       0.67      1.00      0.80         2
   6300000.0       0.00      0.00      0.00         1
   6527820.0       0.00      0.00      0.00         1
   6700000.0       0.00      0.00      0.00         0
   6900000.0       1.00      1.00      1.00         1
   7090000.0       0.00      0.00      0.00         1
   7200000.0       0.00      0.00      0.00         0
   7500000.0       1.00      1.00      1.00         1
   8900000.0       0.00      0.00      0.00         1
   8923600.0       0.00      0.00      0.00         0
   9000000.0       0.33      1.00      0.50         1
   9100000.0       0.00      0.00      0.00         1
   9150000.0       0.00      0.00      0.00         1
   9450000.0       0.00      0.00      0.00         0
   9500000.0       0.00      0.00      0.00         2
  10000000.0       1.00      1.00      1.00         1
  11000000.0       0.33      1.00      0.50         1
  11100000.0       0.00      0.00      0.00         1
  11200000.0       0.00      0.00      0.00         1
  11400000.0       0.00      0.00      0.00         1
  11519074.0       0.00      0.00      0.00         0
  12650000.0       0.00      0.00      0.00         0
  13836170.0       0.00      0.00      0.00         1
  15000000.0       0.00      0.00      0.00         1
  15152514.0       0.00      0.00      0.00         1
  15300000.0       0.00      0.00      0.00         0
  18000000.0       0.00      0.00      0.00         0
  24300000.0       0.00      0.00      0.00         0
  24600000.0       0.00      0.00      0.00         1
  25000000.0       1.00      1.00      1.00         1
  26000000.0       0.50      1.00      0.67         1
  27400000.0       0.00      0.00      0.00         1
  40000000.0       0.00      0.00      0.00         0
  45000000.0       0.00      0.00      0.00         1
  50000000.0       0.00      0.00      0.00         2
  90000000.0       1.00      1.00      1.00         2
  99800000.0       0.00      0.00      0.00         0
 154053900.0       0.00      0.00      0.00         0
 170000000.0       0.00      0.00      0.00         1
 191000000.0       0.00      0.00      0.00         1
 206000000.0       0.00      0.00      0.00         0

    accuracy                           0.99      9888
   macro avg       0.23      0.26      0.24      9888
weighted avg       0.99      0.99      0.99      9888
```

KNeighborsClassifier Model

In [111…

```
# create and fit KNeighborsClassifier model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)

#predict
```

```
pred = knn.predict(X_test)

# #KNN accuracy score
Knn_acc= accuracy_score(y_test, knn.predict(X_test))
print('The accuracy socre using the KNeighborsClassifier is :',Knn_acc)

print(classification_report(y_test, pred))
```

```
The accuracy socre using the KNeighborsClassifier is : 0.9841221682847896
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00      9656
      1700.0       0.00      0.00      0.00         1
      1840.0       0.00      0.00      0.00         0
      2000.0       0.00      0.00      0.00         2
      2200.0       0.00      0.00      0.00         0
      2500.0       0.00      0.00      0.00         1
      3000.0       0.00      0.00      0.00         0
     10000.0       1.00      1.00      1.00         2
     10009.0       0.00      0.00      0.00         1
     12000.0       0.00      0.00      0.00         0
     13000.0       0.00      0.00      0.00         2
     14852.0       0.00      0.00      0.00         1
     15000.0       1.00      1.00      1.00         1
     18000.0       0.00      0.00      0.00         1
     18394.0       0.00      0.00      0.00         1
     20000.0       0.33      1.00      0.50         1
     21466.0       0.00      0.00      0.00         0
     25000.0       0.60      0.43      0.50         7
     26500.0       0.00      0.00      0.00         1
     29000.0       0.00      0.00      0.00         1
     30000.0       0.00      0.00      0.00         2
     31777.0       0.00      0.00      0.00         1
     32521.0       0.00      0.00      0.00         0
     32707.0       0.00      0.00      0.00         1
     33283.0       0.00      0.00      0.00         0
     34060.0       0.00      0.00      0.00         1
     35000.0       0.50      0.50      0.50         2
     39700.0       0.00      0.00      0.00         1
     40000.0       0.67      0.67      0.67         6
     42000.0       0.00      0.00      0.00         0
     43000.0       0.00      0.00      0.00         0
     43823.0       0.00      0.00      0.00         1
     45000.0       0.00      0.00      0.00         1
     45412.0       0.00      0.00      0.00         1
     45457.0       0.00      0.00      0.00         1
     45685.0       0.00      0.00      0.00         0
     47500.0       0.00      0.00      0.00         1
     50000.0       0.50      0.40      0.44         5
     50408.0       0.00      0.00      0.00         1
     60000.0       1.00      1.00      1.00         1
     67000.0       0.00      0.00      0.00         1
     75000.0       0.00      0.00      0.00         3
     86729.0       0.00      0.00      0.00         1
     90000.0       0.00      0.00      0.00         1
     90019.0       0.00      0.00      0.00         1
     95000.0       0.00      0.00      0.00         1
     96325.0       0.00      0.00      0.00         1
     96832.0       0.00      0.00      0.00         0
    100000.0       0.75      0.50      0.60         6
    106536.0       0.00      0.00      0.00         0
    115000.0       0.00      0.00      0.00         1
    120000.0       0.00      0.00      0.00         0
    120941.0       0.00      0.00      0.00         1
    129390.0       0.00      0.00      0.00         0
    130295.0       0.00      0.00      0.00         1
    136700.0       0.00      0.00      0.00         1
    144702.0       0.00      0.00      0.00         0
    145774.0       0.00      0.00      0.00         1
    150000.0       0.50      1.00      0.67         2
    152763.0       0.00      0.00      0.00         0
    154320.0       0.00      0.00      0.00         1
    160922.0       0.00      0.00      0.00         0
    162000.0       0.00      0.00      0.00         1
    170000.0       0.00      0.00      0.00         1
    180000.0       1.00      1.00      1.00         2
    190000.0       0.00      0.00      0.00         1
    196000.0       0.00      0.00      0.00         1
    197634.0       0.00      0.00      0.00         1
    200000.0       0.29      1.00      0.44         2
    203171.0       0.00      0.00      0.00         1
    205000.0       0.00      0.00      0.00         1
    209800.0       0.00      0.00      0.00         0
    213000.0       0.00      0.00      0.00         1
    213094.0       0.00      0.00      0.00         0
    214250.0       0.00      0.00      0.00         1
    220099.0       0.00      0.00      0.00         1
    225195.0       0.00      0.00      0.00         1
```

240000.0	0.00	0.00	0.00	1
245000.0	0.00	0.00	0.00	1
246000.0	0.00	0.00	0.00	1
250000.0	0.75	1.00	0.86	3
253000.0	0.00	0.00	0.00	1
258043.0	0.00	0.00	0.00	1
264000.0	0.00	0.00	0.00	0
270500.0	0.00	0.00	0.00	0
285000.0	0.00	0.00	0.00	1
293114.0	0.00	0.00	0.00	1
300000.0	0.00	0.00	0.00	2
349667.0	0.00	0.00	0.00	1
350000.0	0.50	1.00	0.67	2
352957.0	0.00	0.00	0.00	1
367740.0	0.00	0.00	0.00	0
375000.0	0.00	0.00	0.00	2
378812.0	0.00	0.00	0.00	1
389974.0	0.00	0.00	0.00	0
428000.0	0.00	0.00	0.00	1
450000.0	0.00	0.00	0.00	1
480000.0	0.00	0.00	0.00	0
500000.0	0.71	1.00	0.83	5
522923.0	0.00	0.00	0.00	1
530000.0	0.00	0.00	0.00	1
540000.0	0.00	0.00	0.00	1
545000.0	0.00	0.00	0.00	1
600000.0	1.00	1.00	1.00	1
620000.0	0.00	0.00	0.00	1
628000.0	0.00	0.00	0.00	0
638000.0	0.00	0.00	0.00	1
700000.0	0.50	1.00	0.67	1
704000.0	0.00	0.00	0.00	0
704167.0	0.00	0.00	0.00	0
706000.0	0.00	0.00	0.00	1
726694.0	0.00	0.00	0.00	1
729272.0	0.00	0.00	0.00	0
745000.0	0.00	0.00	0.00	1
750000.0	0.75	1.00	0.86	3
796180.0	0.00	0.00	0.00	1
799430.0	0.00	0.00	0.00	1
800000.0	0.20	1.00	0.33	1
878000.0	0.00	0.00	0.00	1
903915.0	0.00	0.00	0.00	1
943000.0	0.00	0.00	0.00	0
975000.0	0.00	0.00	0.00	1
977000.0	0.00	0.00	0.00	1
1000000.0	0.83	1.00	0.91	5
1150000.0	0.00	0.00	0.00	1
1158093.0	0.00	0.00	0.00	1
1200000.0	1.00	0.40	0.57	5
1400000.0	0.33	0.50	0.40	2
1500000.0	0.67	0.40	0.50	5
1600000.0	1.00	1.00	1.00	1
1690000.0	0.00	0.00	0.00	1
1725000.0	0.00	0.00	0.00	1
2000000.0	0.80	0.40	0.53	10
2100000.0	0.00	0.00	0.00	1
2200000.0	0.00	0.00	0.00	2
2300000.0	1.00	0.50	0.67	2
2440000.0	0.00	0.00	0.00	1
2500000.0	0.50	1.00	0.67	3
2547000.0	0.00	0.00	0.00	1
2600000.0	0.00	0.00	0.00	1
2750000.0	0.00	0.00	0.00	0
2800000.0	0.00	0.00	0.00	1
2814100.0	0.00	0.00	0.00	1
2885317.0	0.00	0.00	0.00	0
3000000.0	1.00	0.67	0.80	6
3500000.0	1.00	1.00	1.00	1
3663934.0	0.00	0.00	0.00	1
3684289.0	0.00	0.00	0.00	1
3900000.0	0.00	0.00	0.00	2
4000000.0	0.00	0.00	0.00	0
4300000.0	0.00	0.00	0.00	0
4500000.0	0.00	0.00	0.00	1
4800000.0	0.00	0.00	0.00	1
4900000.0	0.00	0.00	0.00	1
5000000.0	0.75	1.00	0.86	3
6000000.0	0.50	0.75	0.60	4
6200000.0	0.00	0.00	0.00	2
6300000.0	0.00	0.00	0.00	1
6527820.0	0.00	0.00	0.00	1
6900000.0	0.00	0.00	0.00	1
7000000.0	0.00	0.00	0.00	0
7090000.0	0.00	0.00	0.00	1
7500000.0	0.00	0.00	0.00	1
8858230.0	0.00	0.00	0.00	0
8900000.0	0.00	0.00	0.00	1
9000000.0	0.50	1.00	0.67	1

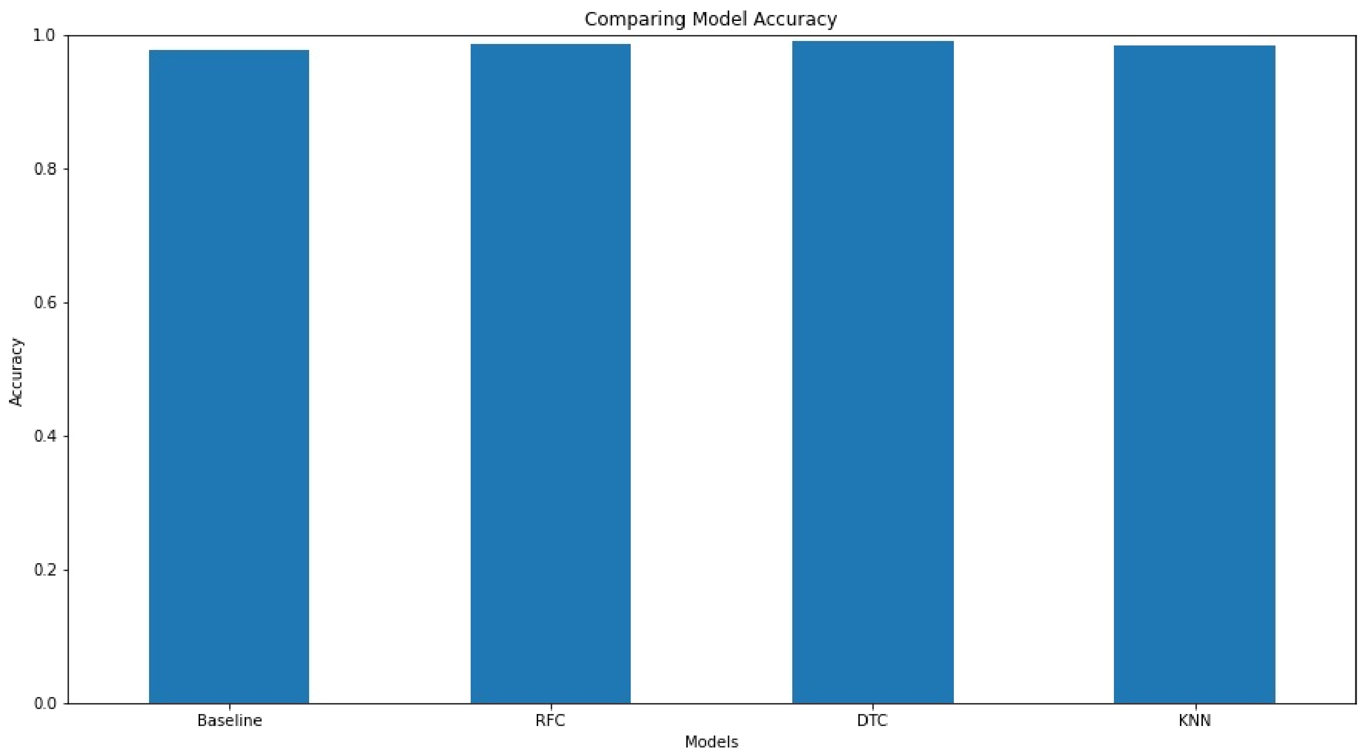
```
   9100000.0       0.00      0.00      0.00         1
   9150000.0       0.00      0.00      0.00         1
   9450000.0       0.00      0.00      0.00         0
   9500000.0       0.00      0.00      0.00         2
  10000000.0       1.00      1.00      1.00         1
  10400000.0       0.00      0.00      0.00         0
  10800000.0       0.00      0.00      0.00         0
  11000000.0       0.00      0.00      0.00         1
  11100000.0       0.00      0.00      0.00         1
  11200000.0       0.00      0.00      0.00         1
  11400000.0       0.00      0.00      0.00         1
  13836170.0       0.00      0.00      0.00         1
  14500000.0       0.00      0.00      0.00         0
  15000000.0       0.00      0.00      0.00         1
  15152514.0       0.00      0.00      0.00         1
  19900000.0       0.00      0.00      0.00         0
  24600000.0       0.00      0.00      0.00         1
  25000000.0       0.33      1.00      0.50         1
  26000000.0       0.00      0.00      0.00         1
  27400000.0       0.00      0.00      0.00         1
  39700000.0       0.00      0.00      0.00         0
  45000000.0       0.00      0.00      0.00         1
  45100000.0       0.00      0.00      0.00         0
  50000000.0       0.00      0.00      0.00         2
  90000000.0       1.00      1.00      1.00         2
 145750000.0       0.00      0.00      0.00         0
 154053900.0       0.00      0.00      0.00         0
 170000000.0       0.00      0.00      0.00         1
 191000000.0       0.00      0.00      0.00         1

    accuracy                           0.98      9888
   macro avg       0.13      0.15      0.13      9888
weighted avg       0.98      0.98      0.98      9888
```

Evaluation

Comparing Model Accuracy

In [112…

```python
pd.DataFrame([baseline_acc, rfc_acc, dtc_acc, Knn_acc]).plot.bar();
plt.xticks(np.arange(4),('Baseline','RFC','DTC','KNN'))
plt.legend().remove()
plt.ylim(0,1)
plt.ylabel('Accuracy')
plt.xlabel('Models')
plt.xticks(rotation = 0)
plt.title('Comparing Model Accuracy');
```



Conclusion / Final Thoughts

We have applied three distinctive machine learning techniques to the data. Decision tree seem to be the most successful out of them. Decision tree accomplished 0.989 accuracy score.which it is perfect and we think we can consider it a successful classification. while baseline accuracy score is 0.976, KNN accuracy score is 0.984, Random Forest accuracy score is 0.985.

In [ ]:

```

```