This jupyter notebook is prepared by “Daniel Rodriguez”.

1. Run the block below to upload the dataset. (Note that the file list gets refreshed every time your runtime is disconnected. Simply run this when you return to upload the file again using the files API. Once you run, it should wait for you to upload the file. (1pt)

```
from google.colab import files
uploaded = files.upload()
```

Choose Files startup_info_.csv

- **startup_info_.csv**(text/csv) - 168764 bytes, last modified: 2/10/2023 - 100% done

```
Saving startup_info_.csv to startup_info_ (6).csv
```

2. Import numpy, pandas, matplotlib.pyplot and seaborn packages. (2pt)

If you need additional packages, you can import it on the go in any code-block below.

```
#TODO
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

3. Import the dataset into a pandas dataframe. Then report how many rows and columns are present in the dataset. (2pt)

```
#TODO
df = pd.read_csv('startup_info_.csv')
df.shape
```

```
(923, 28)
```

4. Call the describe method to see summary statistics of the numerical attribute columns. (1pt)

```
#TODO
df.describe()
```

	Unnamed: 0	latitude	longitude	labels	age_first_funding_year	age
count	923.000000	923.000000	923.000000	923.000000	923.000000	
mean	572.297941	38.517442	-103.539212	0.646804	2.235630	
std	333.585431	3.741497	22.394167	0.478222	2.510449	
min	1.000000	25.752358	-122.756956	0.000000	-9.046600	
25%	283.500000	37.388869	-122.198732	0.000000	0.576700	
50%	577.000000	37.779281	-118.374037	1.000000	1.446600	
75%	866.500000	40.730646	-77.214731	1.000000	3.575350	
max	1153.000000	59.335232	18.057121	1.000000	21.895900	

5.1 List all attribute columns (1pt)

```
#TODO
list(df.columns)
```

```
['Unnamed: 0',
 'state_code',
 'latitude',
```

```
 'longitude',
 'zip_code',
 'id',
 'city',
 'Unnamed: 6',
 'name',
 'labels',
 'founded_at',
 'closed_at',
 'first_funding_at',
 'last_funding_at',
 'age_first_funding_year',
 'age_last_funding_year',
 'age_first_milestone_year',
 'age_last_milestone_year',
 'relationships',
 'funding_rounds',
 'funding_total_usd',
 'milestones',
 'state_code.1',
 'category_code',
 'object_id',
 'avg_participants',
 'is_top500',
 'status']
```

5.2 The "Unnamed: 0","Unnamed: 6", "state_code.1" and "object_id" feature columns are not useful. Drop them in-place. (1pt)

```
#TODO
df.drop(columns=['Unnamed: 0','Unnamed: 6','state_code.1','object_id'], inplace=True)
```

6.1 Show all the numeric columns and save it to a new dataframe. (2pt)

```
#TODO
print(df.select_dtypes(include=np.number))
numDf = df.select_dtypes(include=np.number)
```

```
      latitude   longitude  labels  age_first_funding_year  \
0    42.358880  -71.056820       1                  2.2493
1    37.238916 -121.973718       1                  5.1260
2    32.901049 -117.192656       1                  1.0329
3    37.320309 -122.050040       1                  3.1315
4    37.779281 -122.419236       0                  0.0000
..         ...         ...     ...                     ...
918  37.740594 -122.376471       1                  0.5178
919  42.504817  -71.195611       0                  7.2521
920  37.408261 -122.015920       0                  8.4959
921  37.556732 -122.288378       1                  0.7589
922  37.386778 -121.966277       1                  3.1205

     age_last_funding_year  age_first_milestone_year  age_last_milestone_year  \
0                   3.0027                    4.6685                   6.7041
1                   9.9973                    7.0055                   7.0055
2                   1.0329                    1.4575                   2.2055
3                   5.3151                    6.0027                   6.0027
4                   1.6685                    0.0384                   0.0384
..                     ...                       ...                      ...
918                 0.5178                    0.5808                   4.5260
919                 9.2274                    6.0027                   6.0027
920                 8.4959                    9.0055                   9.0055
921                 2.8329                    0.7589                   3.8356
922                 3.1205                    4.0027                   4.0027

     relationships  funding_rounds  funding_total_usd  milestones  \
0                3               3             375000           3
1                9               4           40100000           1
2                5               1            2600000           2
3                5               3           40000000           1
4                2               2            1300000           1
..             ...             ...                ...         ...
918              9               1            1100000           2
919              1               3           52000000           1
920              5               1           44000000           1
921             12               2           15500000           2
922              4               1           20000000           1

     avg_participants  is_top500
0              1.0000          0
```

```
1              4.7500          1
2              4.0000          1
3              3.3333          1
4              1.0000          1
..                ...        ...
918            6.0000          1
919            2.6667          1
920            8.0000          1
921            1.0000          1
922            3.0000          1

[923 rows x 13 columns]
```

6.2 Plot distributions of the numeric columns using histogram and record the skew of each distribution. (Note: positive value = right skewed, negative value = left skewed) (4pt)

```
#TODO
hist = numDf.hist(figsize=(15,15))
for i in numDf.columns:
  if numDf[i].skew() > 0:
    distribution = 'right skewed (positive)'
  elif numDf[i].skew() < 0:
    distribution = 'left skewed (negative)'
  else:
    distribution = 'symmetrical'
  print(i+' is '+ distribution)
```

```
latitude is right skewed (positive)
longitude is right skewed (positive)
labels is left skewed (negative)
age_first_funding_year is right skewed (positive)
age_last_funding_year is right skewed (positive)
age_first_milestone_year is right skewed (positive)
age_last_milestone_year is right skewed (positive)
relationships is right skewed (positive)
funding_rounds is right skewed (positive)
```

7. Show all the categorical columns and save it to a new dataframe. (2pt)

```
#TODO
print(df.select_dtypes(include=object))
catDf = df.select_dtypes(include=object)
```

```
    state_code zip_code       id           city                 name  \
0           CA    92101   c:6669      San Diego          Bandsintown
1           CA    95032  c:16283      Los Gatos            TriCipher
2           CA    92121  c:65620      San Diego                Plixi
3           CA    95014  c:42668      Cupertino     Solidcore Systems
4           CA    94105  c:65806  San Francisco       Inhale Digital
..         ...      ...      ...            ...                  ...
918         CA    94107  c:21343  San Francisco              CoTweet
919         MA     1803  c:41747     Burlington   Reef Point Systems
920         CA    94089  c:31549      Sunnyvale      Paracor Medical
921         CA    94404  c:33198  San Francisco              Causata
922         CA    95054  c:26702    Santa Clara  Asempra Technologies

    founded_at  closed_at first_funding_at last_funding_at category_code  \
0     1/1/2007        NaN         4/1/2009        1/1/2010         music
1     1/1/2000        NaN        2/14/2005      12/28/2009    enterprise
2    3/18/2009        NaN        3/30/2010       3/30/2010           web
3     1/1/2002        NaN        2/17/2005       4/25/2007      software
4     8/1/2010  10/1/2012         8/1/2010        4/1/2012   games_video
..         ...        ...              ...             ...           ...
918   1/1/2009        NaN         7/9/2009        7/9/2009   advertising
919   1/1/1998  6/25/2008         4/1/2005       3/23/2007      security
920   1/1/1999  6/17/2012        6/29/2007       6/29/2007       biotech
921   1/1/2009        NaN        10/5/2009       11/1/2011      software
922   1/1/2003        NaN        2/13/2006       2/13/2006      security

       status
0    acquired
1    acquired
2    acquired
3    acquired
4      closed
..        ...
918  acquired
919    closed
920    closed
921  acquired
922  acquired

[923 rows x 11 columns]
```

8. Examine missing values (2+2+3=7pt)

8.1 Show a list with column wise count of missing values and display the list in count wise descending order.

```
#TODO
df.isna().sum().sort_values(ascending=False)
```

```
closed_at                   588
age_last_milestone_year     152
age_first_milestone_year    152
state_code                    0
age_last_funding_year         0
is_top500                     0
avg_participants              0
category_code                 0
milestones                    0
funding_total_usd             0
funding_rounds                0
relationships                 0
age_first_funding_year        0
latitude                      0
last_funding_at               0
first_funding_at              0
```

```
founded_at                0
labels                    0
name                      0
city                      0
id                        0
zip_code                  0
longitude                 0
status                    0
dtype: int64
```

8.2 Show columnwise percentage of missing values.

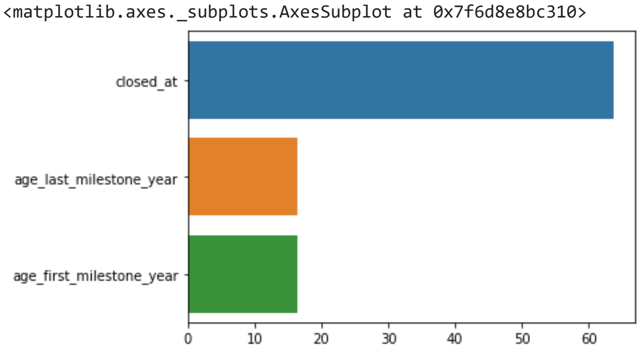
```
#TODO
df.isnull().mean().sort_values(ascending=False) * 100
```

```
closed_at                 63.705309
age_last_milestone_year   16.468039
age_first_milestone_year  16.468039
state_code                 0.000000
age_last_funding_year      0.000000
is_top500                  0.000000
avg_participants           0.000000
category_code              0.000000
milestones                 0.000000
funding_total_usd          0.000000
funding_rounds             0.000000
relationships              0.000000
age_first_funding_year     0.000000
latitude                   0.000000
last_funding_at            0.000000
first_funding_at           0.000000
founded_at                 0.000000
labels                     0.000000
name                       0.000000
city                       0.000000
id                         0.000000
zip_code                   0.000000
longitude                  0.000000
status                     0.000000
dtype: float64
```

8.3 Display a bar plot to visualize only the columns with missing values and their percentage count.

```
#TODO
perc = df.isnull().mean().sort_values(ascending=False) * 100
perc = perc.where(perc > 1).dropna()
sns.barplot(perc,x=perc.values,y=perc.index)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d8e8bc310>
```



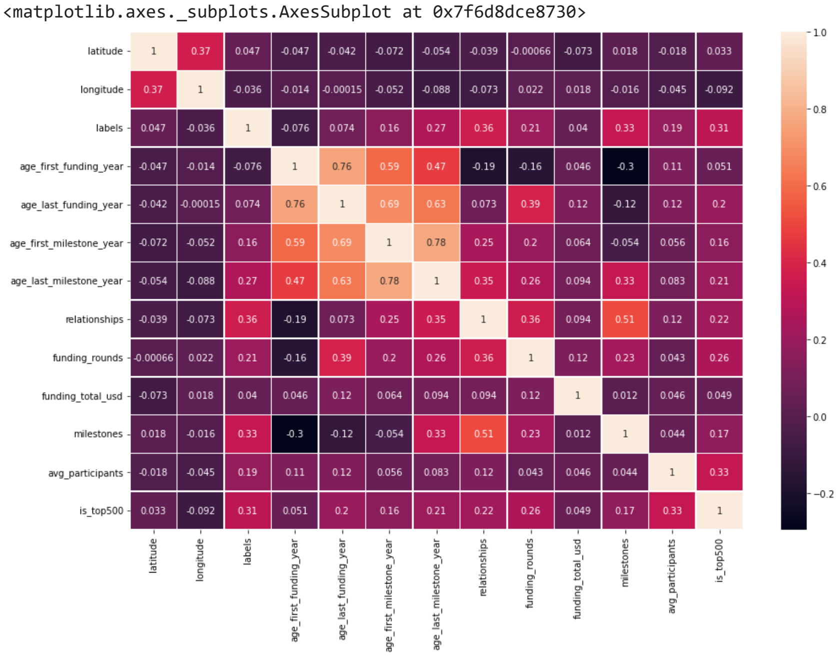
9. Label Encoding : Copy the dataframe to a new one. Then using scikitlearn's Label Encoder, transform the "status" column to 0-1. (5pt)

```
#TODO
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
encDf = df
encDf['status'] = le.fit_transform(encDf['status'])
```

10. Correlation: Use seaborn's heatmap to visualize the correlation between numeric features. (3pt)

```
#TODO
plt.figure(figsize = (15,10))
sns.heatmap(numDf.corr(),linewidth=.5, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d8dce8730>
```



11.1 Use seaborn's countplot to visualize relationship between "*state_code*" and "*labels*". Comment on which state produced majority of successful startups (3pt)

```
#TODO
plt.figure(figsize = (15,10))
sns.countplot(data=df,x='state_code',hue='labels')
print('state_code CA had the most successful startups based on the chart below:\n')
```

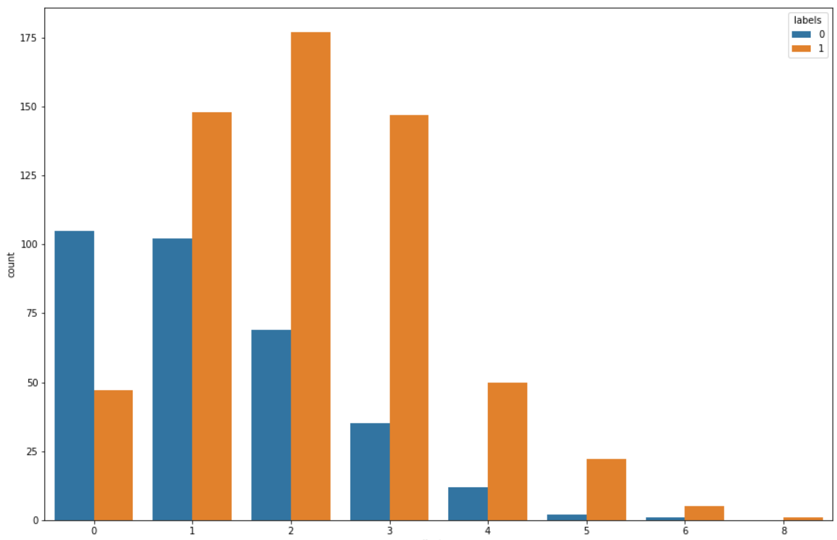
```
state_code CA had the most successful startups based on the chart below:
```

11.2 Use seaborn's countplot to visualize relationship between "*milestones*" and "*labels*". Comment on which milestone made the statistically highest number of successful startups (3pt)

Double-click (or enter) to edit

```
#TODO
plt.figure(figsize = (15,10))
sns.countplot(data=df,x='milestones',hue='labels')
print('milestones 2 had the most successful startups based on the chart below:\n')
```

```
milestones 2 had the most successful startups based on the chart below:
```



12. Drop features with duplicate values in-place, then show dataframe's new shape. (1pt)

```
#TODO
df.drop_duplicates(inplace=True)
df.shape
```

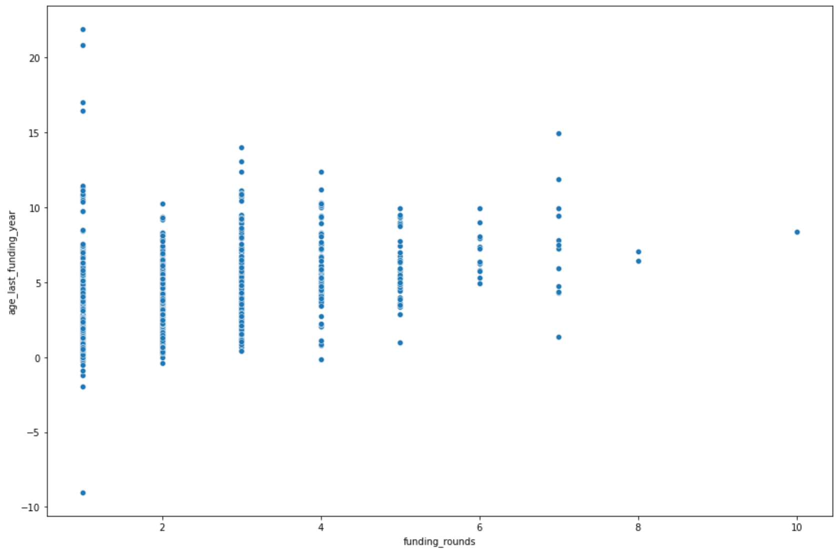
```
(923, 24)
```

13. From correlation heatmap above, comment on which feature has the highest correlation with "*funding_rounds*". Visualize a scatterplot with that and "*funding_rounds*". (3+3 = 6pt)

```
#TODO
print('According to the heatmap above age_last_funding_year appeared to have the highest correlation with funding_rounds')
```

```
plt.figure(figsize = (15,10))
sns.scatterplot(data=numDf, x='funding_rounds',y='age_last_funding_year')
```

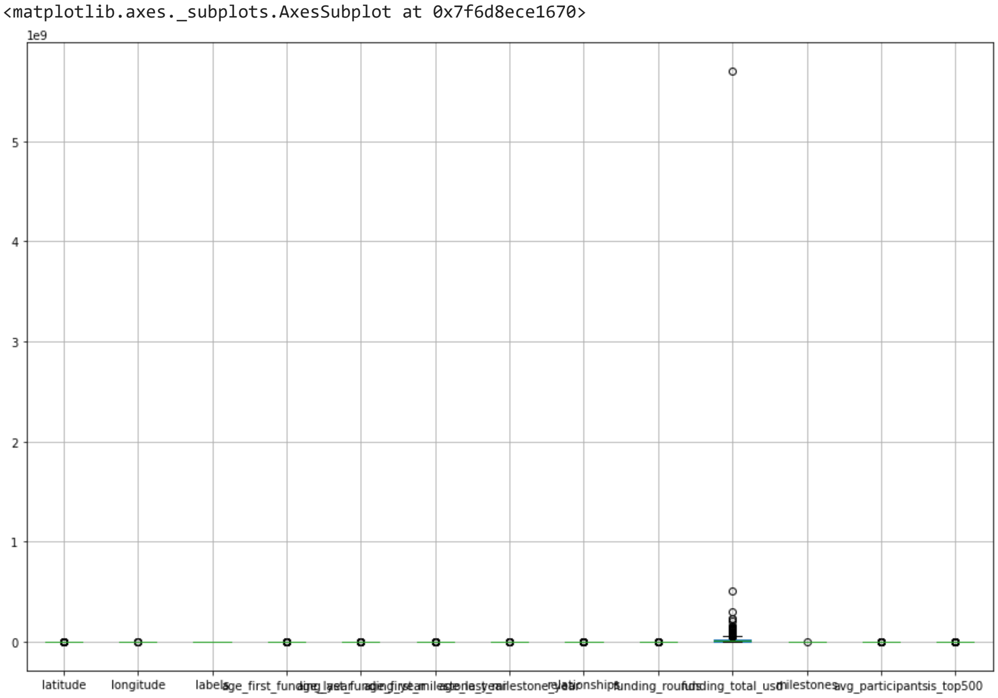
```
According to the heatmap above age_last_funding_year appeared to have the highest
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d8f4662e0>
```



14. Show boxplots for the numeric features to detect outliers. (4pt)

```
#TODO
numDf.boxplot(figsize=(15,10))
```

```
/usr/local/lib/python3.8/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDeprecationWarni
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d8ece1670>
```



15. Summary and Discussion: Mention what additional steps are required to use this dataset in a binary classifier. Eg: any column to remove, any record to remove, any distribution to rebalance, any features to be joined together to generate new feature etc. (2pt)

```
from sklearn.linear_model import LogisticRegression
encDf['state_code'] = le.fit_transform(encDf['state_code'])
encDf['city'] = le.fit_transform(encDf['city'])
encDf['category_code'] = le.fit_transform(encDf['category_code'])
encDf['zip_code'] = le.fit_transform(encDf['zip_code'])
encDf = encDf.dropna()

x = encDf.loc[:, ~encDf.columns.isin(['id', 'name','founded_at','closed_at','first_funding_at','last_funding_at'])]
y = encDf['labels']

clf = LogisticRegression().fit(x,y)
```

TODO

In order to use this for binary classifier we would begin with encoding all of the string columns inclduing state_code, city, category_code, zip_code. Dropping the date and identifier columns including id, name, founded_at, closed_at, first_funding_at, last_funding_at, and potentially zip_code. Following that you would drop all NaN values in the dataframe. After then you would begin running new diagnostics on the now transformed data. These diagnostics would include feature selection using various methods such as p-values, fitting a model and seeing the most relevent features based on scores. Diagnostics would also include checking for relevant interactions, those can be found either by deductive reasoning and testing or by forcing interaction_only model fits.