Project Report: Distributed Computing (CDCSC15)

Submitted By:-

1. Vansh Aggarwal - 2021UCD2109
2. Vaibhav Yadav - 2021UCD2139
3. Pulkit Aggarwal - 2021UCD2152
4. Varun Sanjeevan - 2021UCD2169

Submitted To:- Dr. MPS Bhatia

INDEX

Startup Success Prediction

Can we predict if a start-up will succeed or fail?

Abstract

A startup or start-up is a company or project begun by an entrepreneur to seek, develop, and validate a scalable economic model. While entrepreneurship refers to all new businesses, including self-employment and businesses that never intend to become registered, startups refer to new businesses that intend to grow large beyond the solo founder. Startups face high uncertainty and have high rates of failure, but a minority of them do go on to be successful and influential. Some startups become unicorns: privately held startup companies valued at over US$1 billion.

Startups play a major role in economic growth. They bring new ideas, spur innovation, create employment thereby moving the economy. There has been an exponential growth in startups over the past few years. Predicting the success of a startup allows investors to find companies that have the potential for rapid growth, thereby allowing them to be one step ahead of the competition.

Objective

The objective is to predict whether a startup which is currently operating turns into a success or a failure. The success of a company is defined as the event that gives the company's founders a large sum of money through the process of M&A (Merger and Acquisition) or an IPO (Initial Public Offering). A company would be considered as failed if it had to be shut down.

About the Data

The data contains industry trends, investment insights and individual company information. There are 48 columns/features. Some of the features are:

- age_first_funding_year – quantitative
- age_last_funding_year – quantitative
- relationships – quantitative
- funding_rounds – quantitative
- funding_total_usd – quantitative
- milestones – quantitative

- age_first_milestone_year – quantitative
- age_last_milestone_year – quantitative
- state – categorical
- industry_type – categorical
- has_VC – categorical
- has_angel – categorical
- has_roundA – categorical
- has_roundB – categorical
- has_roundC – categorical
- has_roundD – categorical
- avg_participants – quantitative
- is_top500 – categorical
- status(acquired/closed) – categorical (the target variable, if a startup is ‘acquired’ by some other organisation, means the startup succeed)

Methodology:

1. Data Preprocessing:
    - Data Collection: Initially, a dataset of different startups across the USA was collected. This dataset likely includes various attributes about each startup, such as funding history, location, industry, and other relevant information.
    - Data Cleaning: The collected data might contain noise, missing values, or outliers. Data cleaning involves handling missing values by either imputing them or removing rows with missing data. Outliers might be treated or retained based on the specific characteristics of the data.
    - Feature Engineering: This step involves creating new features or transforming existing ones to better represent the information in the dataset. For example, you might create new features like the age of the startup, total funding raised, or location-based features.
    - Data Encoding: Categorical variables, such as industry type or location, need to be encoded into a numerical format for machine learning algorithms to work. Common techniques include one-hot encoding or label encoding.
    - Data Splitting: The dataset is typically divided into training and testing sets. The training set is used to train the machine learning models, while the testing set is used to evaluate their performance.

2. Machine Learning Models:

    In this project, two popular machine learning algorithms were employed:

    - Decision Tree: A Decision Tree is a tree-like model that makes decisions based on a series of rules learned from the data. It is a straightforward and interpretable model that can handle both categorical and numerical data.

Decision Trees are prone to overfitting, which means they can perform very well on the training data but poorly on unseen data.

- Random Forest: Random Forest is an ensemble learning method that builds multiple Decision Trees and combines their predictions. It reduces overfitting compared to a single Decision Tree and generally provides more robust predictions. Random Forest can handle a large number of features and is less sensitive to hyperparameter tuning.

3. Frontend Development:
    - To provide a user-friendly interface for making predictions, we created a simple frontend using the Gradio library in Python.
    - The frontend allowed users to input startup information and receive predictions based on the trained machine learning models.

Algorithms Used:

In this project, two popular machine learning algorithms were employed:

- **Decision Tree:** A Decision Tree is a tree-like model that makes decisions based on a series of rules learned from the data. It is a straightforward andinterpretable model that can handle both categorical and numerical data. Decision Trees are prone to overfitting, which means they can perform verywell on the training data but poorly on unseen data.

- **Random Forest:** Random Forest is an ensemble learning method that builds multiple Decision Trees and combines their predictions. It reduces overfittingcompared to a single Decision Tree and generally provides more robust predictions. Random Forest can handle a large number of features and is less sensitive to hyperparameter tuning.

Source Code:

```
!pip install -q gradio
import gradio as gr
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import graphviz
from IPython.display import display, Image
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics
from sklearn.metrics import accuracy_score, classification_report
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns',None)
```

```
df=pd.read_csv("/content/startup data.csv")
df
```

	Unnamed: 0	state_code	latitude	longitude	zip_code	id	city	Unnamed: 6	name	labels	founded_at	closed_at
0	1005	CA	42.358880	-71.056820	92101	c:6669	San Diego	NaN	Bandsintown	1	1/1/2007	NaN
1	204	CA	37.238916	-121.973718	95032	c:16283	Los Gatos	NaN	TriCipher	1	1/1/2000	NaN
2	1001	CA	32.901049	-117.192656	92121	c:65620	San Diego	San Diego CA 92121	Plixi	1	3/18/2009	NaN
3	738	CA	37.320309	-122.050040	95014	c:42668	Cupertino	Cupertino CA 95014	Solidcore Systems	1	1/1/2002	NaN
4	1002	CA	37.779281	-122.419236	94105	c:65806	San Francisco	San Francisco CA 94105	Inhale Digital	0	8/1/2010	10/1/2012
...	...	...	...	...	...	...	...	...	...	...	...	...
918	352	CA	37.740594	-122.376471	94107	c:21343	San Francisco	NaN	CoTweet	1	1/1/2009	NaN
919	721	MA	42.504817	-71.195611	1803	c:41747	Burlington	Burlington MA 1803	Reef Point Systems	0	1/1/1998	6/25/2008
920	557	CA	37.408261	-122.015920	94089	c:31549	Sunnyvale	NaN	Paracor Medical	0	1/1/1999	6/17/2012
921	589	CA	37.556732	-122.288378	94404	c:33198	San Francisco	NaN	Causata	1	1/1/2009	NaN
922	462	CA	37.386778	-121.966277	95054	c:26702	Santa Clara	Santa Clara CA 95054	Asempra Technologies	1	1/1/2003	NaN

923 rows × 49 columns

```
print("Dataset Shape (Rows, Column) : ",df.shape)
print("Size of Dataset = ",df.size)
```

```
Dataset Shape (Rows, Column) :  (923, 49)
Size of Dataset =  45227
```

```
print("Columns : \n",df.columns)
```

```
Columns :
 Index(['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', 'is_CA', 'is_NY',
       'is_MA', 'is_TX', 'is_otherstate', 'category_code', 'is_software',
       'is_web', 'is_mobile', 'is_enterprise', 'is_advertising',
       'is_gamesvideo', 'is_ecommerce', 'is_biotech', 'is_consulting',
       'is_othercategory', 'object_id', 'has_VC', 'has_angel', 'has_roundA',
       'has_roundB', 'has_roundC', 'has_roundD', 'avg_participants',
       'is_top500', 'status'],
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 923 entries, 0 to 922
Data columns (total 49 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                923 non-null    int64
 1   state_code                923 non-null    object
 2   latitude                  923 non-null    float64
 3   longitude                 923 non-null    float64
 4   zip_code                  923 non-null    object
 5   id                        923 non-null    object
 6   city                      923 non-null    object
 7   Unnamed: 6                430 non-null    object
 8   name                      923 non-null    object
 9   labels                    923 non-null    int64
 10  founded_at                923 non-null    object
 11  closed_at                 335 non-null    object
 12  first_funding_at          923 non-null    object
 13  last_funding_at           923 non-null    object
 14  age_first_funding_year    923 non-null    float64
 15  age_last_funding_year     923 non-null    float64
 16  age_first_milestone_year  771 non-null    float64
 17  age_last_milestone_year   771 non-null    float64
 18  relationships             923 non-null    int64
 19  funding_rounds            923 non-null    int64
 20  funding_total_usd         923 non-null    int64
 21  milestones                923 non-null    int64
 22  state_code.1              922 non-null    object
 23  is_CA                     923 non-null    int64
 24  is_NY                     923 non-null    int64
 25  is_MA                     923 non-null    int64
 26  is_TX                     923 non-null    int64
 27  is_otherstate             923 non-null    int64
 28  category_code             923 non-null    object
 29  is_software               923 non-null    int64
 30  is_web                    923 non-null    int64
 31  is_mobile                 923 non-null    int64
 32  is_enterprise             923 non-null    int64
 33  is_advertising            923 non-null    int64
 34  is_gamesvideo             923 non-null    int64
 35  is_ecommerce              923 non-null    int64
 36  is_biotech                923 non-null    int64
 37  is_consulting             923 non-null    int64
 38  is_othercategory          923 non-null    int64
 39  object_id                 923 non-null    object
 40  has_VC                    923 non-null    int64
 41  has_angel                 923 non-null    int64
 42  has_roundA                923 non-null    int64
 43  has_roundB                923 non-null    int64
 44  has_roundC                923 non-null    int64
 45  has_roundD                923 non-null    int64
 46  avg_participants          923 non-null    float64
 47  is_top500                 923 non-null    int64
 48  status                    923 non-null    object
dtypes: float64(7), int64(28), object(14)
memory usage: 353.5+ KB
```

```
df.describe()
```

	Unnamed: 0	latitude	longitude	labels	age_first_funding_year	age_la
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	

```
df_cat=df.select_dtypes(include='object')
df_cat
```

	state_code	zip_code	id	city	Unnamed: 6	name	founded_at	clos
0	CA	92101	c:6669	San Diego	NaN	Bandsintown	1/1/2007	
1	CA	95032	c:16283	Los Gatos	NaN	TriCipher	1/1/2000	
2	CA	92121	c:65620	San Diego	San Diego CA 92121	Plixi	3/18/2009	
3	CA	95014	c:42668	Cupertino	Cupertino CA 95014	Solidcore Systems	1/1/2002	
4	CA	94105	c:65806	San Francisco	San Francisco CA 94105	Inhale Digital	8/1/2010	10/1
...	...	...	...	...	...	...	...	
918	CA	94107	c:21343	San Francisco	NaN	CoTweet	1/1/2009	
919	MA	1803	c:41747	Burlington	Burlington MA 1803	Reef Point Systems	1/1/1998	6/25
920	CA	94089	c:31549	Sunnyvale	NaN	Paracor	1/1/1999	6/17

```
numeric=['int8', 'int16', 'int32', 'int64', 'float16', 'float32', 'float64']
df_num=df.select_dtypes(include=numeric)
df_num
```

	Unnamed: 0	latitude	longitude	labels	age_first_funding_year	age_last_fundin
0	1005	42.358880	-71.056820	1	2.2493	
1	204	37.238916	-121.973718	1	5.1260	
2	1001	32.901049	-117.192656	1	1.0329	
3	738	37.320309	-122.050040	1	3.1315	
4	1002	37.779281	-122.419236	0	0.0000	
...	...	...	...	...	...	
918	352	37.740594	-122.376471	1	0.5178	
919	721	42.504817	-71.195611	0	7.2521	
920	557	37.408261	-122.015920	0	8.4959	
921	589	37.556732	-122.288378	1	0.7589	
922	462	37.386778	-121.966277	1	3.1205	

923 rows × 35 columns

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

```
Unnamed: 0                   0
state_code                   0
latitude                     0
longitude                    0
zip_code                     0
id                           0
city                         0
Unnamed: 6                 493
name                         0
labels                       0
founded_at                   0
closed_at                  588
first_funding_at             0
last_funding_at              0
age_first_funding_year       0
age_last_funding_year        0
age_first_milestone_year   152
age_last_milestone_year    152
relationships                0
funding_rounds               0
funding_total_usd            0
milestones                   0
state_code.1                 1
is_CA                        0
is_NY                        0
is_MA                        0
is_TX                        0
is_otherstate                0
category_code                0
```

```
is_software                 0
is_web                      0
is_mobile                   0
is_enterprise               0
is_advertising              0
is_gamesvideo               0
is_ecommerce                0
is_biotech                  0
is_consulting               0
is_othercategory            0
object_id                   0
has_VC                      0
has_angel                   0
has_roundA                  0
has_roundB                  0
has_roundC                  0
has_roundD                  0
avg_participants            0
is_top500                   0
status                      0
dtype: int64
```

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

```
Unnamed: 0                  0
state_code                  0
latitude                    0
longitude                   0
zip_code                    0
id                          0
city                        0
Unnamed: 6                493
name                        0
labels                      0
founded_at                  0
closed_at                 588
first_funding_at            0
last_funding_at             0
age_first_funding_year      0
age_last_funding_year       0
age_first_milestone_year  152
age_last_milestone_year   152
relationships               0
funding_rounds              0
funding_total_usd           0
milestones                  0
state_code.1                1
is_CA                       0
is_NY                       0
is_MA                       0
is_TX                       0
is_otherstate               0
category_code               0
is_software                 0
is_web                      0
is_mobile                   0
is_enterprise               0
is_advertising              0
is_gamesvideo               0
is_ecommerce                0
is_biotech                  0
is_consulting               0
is_othercategory            0
object_id                   0
has_VC                      0
has_angel                   0
has_roundA                  0
has_roundB                  0
has_roundC                  0
has_roundD                  0
avg_participants            0
is_top500                   0
status                      0
dtype: int64
```

```
columns=df.columns
d_c1=[]
for i in columns:
  df1=df[i].isnull().sum()
  r,c=df.shape
  val=(df1/r)*100
  if val >= 50:
    d_c1.append(i)

print(d_c1)
```

```
['Unnamed: 6', 'closed_at']
```

```
df.drop(['Unnamed: 6','closed_at'],axis=1,inplace=True)
```

```
columns=df.columns
d_c2=[]
for i in columns:
  df2=df[i].isna().sum()
  r,c=df.shape
  val=(df1/r)*100
  if val >= 50:
    d_c2.append(i)

print(d_c2)
```

```
[]
```

```
mean_value1=df['age_first_milestone_year'].mean()
mean_value2=df['age_last_milestone_year'].mean()
df["age_first_milestone_year"].fillna(value=mean_value1,inplace=True)
df["age_last_milestone_year"].fillna(value=mean_value2,inplace=True)
```

```
n=df[df['state_code.1'].isna()==True].index.item()
df.drop(n,axis=0,inplace=True)
```

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

```
Unnamed: 0                  0
state_code                  0
latitude                    0
longitude                   0
zip_code                    0
id                          0
city                        0
name                        0
labels                      0
founded_at                  0
first_funding_at            0
last_funding_at             0
age_first_funding_year      0
age_last_funding_year       0
age_first_milestone_year    0
age_last_milestone_year     0
relationships               0
funding_rounds              0
funding_total_usd           0
milestones                  0
state_code.1                0
is_CA                       0
is_NY                       0
is_MA                       0
is_TX                       0
is_otherstate               0
category_code               0
is_software                 0
is_web                      0
is_mobile                   0
is_enterprise               0
is_advertising              0
is_gamesvideo               0
is_ecommerce                0
is_biotech                  0
is_consulting               0
is_othercategory            0
object_id                   0
has_VC                      0
has_angel                   0
has_roundA                  0
has_roundB                  0
has_roundC                  0
has_roundD                  0
avg_participants            0
is_top500                   0
status                      0
dtype: int64
```

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

```
Unnamed: 0                  0
state_code                  0
latitude                    0
longitude                   0
zip_code                    0
id                          0
city                        0
```

```
name                        0
labels                      0
founded_at                  0
first_funding_at            0
last_funding_at             0
age_first_funding_year      0
age_last_funding_year       0
age_first_milestone_year    0
age_last_milestone_year     0
relationships               0
funding_rounds              0
funding_total_usd           0
milestones                  0
state_code.1                0
is_CA                       0
is_NY                       0
is_MA                       0
is_TX                       0
is_otherstate               0
category_code               0
is_software                 0
is_web                      0
is_mobile                   0
is_enterprise               0
is_advertising              0
is_gamesvideo               0
is_ecommerce                0
is_biotech                  0
is_consulting               0
is_othercategory            0
object_id                   0
has_VC                      0
has_angel                   0
has_roundA                  0
has_roundB                  0
has_roundC                  0
has_roundD                  0
avg_participants            0
is_top500                   0
status                      0
dtype: int64
```

```
df[df.duplicated()]
```

Unnamed: 0	state_code	latitude	longitude	zip_code	id	city	name	labels	founded

```
num_columns=df_num.columns
print(num_columns)
```

```
Index(['Unnamed: 0', 'latitude', 'longitude', 'labels',
       'age_first_funding_year', 'age_last_funding_year',
       'age_first_milestone_year', 'age_last_milestone_year', 'relationships',
       'funding_rounds', 'funding_total_usd', 'milestones', 'is_CA', 'is_NY',
       'is_MA', 'is_TX', 'is_otherstate', 'is_software', 'is_web', 'is_mobile',
       'is_enterprise', 'is_advertising', 'is_gamesvideo', 'is_ecommerce',
       'is_biotech', 'is_consulting', 'is_othercategory', 'has_VC',
       'has_angel', 'has_roundA', 'has_roundB', 'has_roundC', 'has_roundD',
       'avg_participants', 'is_top500'],
      dtype='object')
```

```
for a in range(len(num_columns)):
  if num_columns[a]=="latitude" or num_columns[a]=="longitude":
    pass
  else:
    print("Is there any negative value in '{}' column : {} ".
          format(num_columns[a],(df[num_columns[a]]<0).any()))
```

```
Is there any negative value in 'Unnamed: 0' column : False
Is there any negative value in 'labels' column : False
Is there any negative value in 'age_first_funding_year' column : True
Is there any negative value in 'age_last_funding_year' column : True
Is there any negative value in 'age_first_milestone_year' column : True
Is there any negative value in 'age_last_milestone_year' column : True
Is there any negative value in 'relationships' column : False
Is there any negative value in 'funding_rounds' column : False
Is there any negative value in 'funding_total_usd' column : False
Is there any negative value in 'milestones' column : False
Is there any negative value in 'is_CA' column : False
Is there any negative value in 'is_NY' column : False
Is there any negative value in 'is_MA' column : False
Is there any negative value in 'is_TX' column : False
Is there any negative value in 'is_otherstate' column : False
Is there any negative value in 'is_software' column : False
```

```
Is there any negative value in 'is_web' column : False
Is there any negative value in 'is_mobile' column : False
Is there any negative value in 'is_enterprise' column : False
Is there any negative value in 'is_advertising' column : False
Is there any negative value in 'is_gamesvideo' column : False
Is there any negative value in 'is_ecommerce' column : False
Is there any negative value in 'is_biotech' column : False
Is there any negative value in 'is_consulting' column : False
Is there any negative value in 'is_othercategory' column : False
Is there any negative value in 'has_VC' column : False
Is there any negative value in 'has_angel' column : False
Is there any negative value in 'has_roundA' column : False
Is there any negative value in 'has_roundB' column : False
Is there any negative value in 'has_roundC' column : False
Is there any negative value in 'has_roundD' column : False
Is there any negative value in 'avg_participants' column : False
Is there any negative value in 'is_top500' column : False
```

```
df=df.drop(df[df.age_first_funding_year<0].index)
df=df.drop(df[df.age_last_funding_year<0].index)
df=df.drop(df[df.age_first_milestone_year<0].index)
df=df.drop(df[df.age_last_milestone_year<0].index)
```

```
for a in range(len(num_columns)):
  if num_columns[a]=="latitude" or num_columns[a]=="longitude":
    pass
  else:
    print("Is there any negative value in '{}' column : {} ".
          format(num_columns[a],(df[num_columns[a]]<0).any()))
```

```
Is there any negative value in 'Unnamed: 0' column : False
Is there any negative value in 'labels' column : False
Is there any negative value in 'age_first_funding_year' column : False
Is there any negative value in 'age_last_funding_year' column : False
Is there any negative value in 'age_first_milestone_year' column : False
Is there any negative value in 'age_last_milestone_year' column : False
Is there any negative value in 'relationships' column : False
Is there any negative value in 'funding_rounds' column : False
Is there any negative value in 'funding_total_usd' column : False
Is there any negative value in 'milestones' column : False
Is there any negative value in 'is_CA' column : False
Is there any negative value in 'is_NY' column : False
Is there any negative value in 'is_MA' column : False
Is there any negative value in 'is_TX' column : False
Is there any negative value in 'is_otherstate' column : False
Is there any negative value in 'is_software' column : False
Is there any negative value in 'is_web' column : False
Is there any negative value in 'is_mobile' column : False
Is there any negative value in 'is_enterprise' column : False
Is there any negative value in 'is_advertising' column : False
Is there any negative value in 'is_gamesvideo' column : False
Is there any negative value in 'is_ecommerce' column : False
Is there any negative value in 'is_biotech' column : False
Is there any negative value in 'is_consulting' column : False
Is there any negative value in 'is_othercategory' column : False
Is there any negative value in 'has_VC' column : False
Is there any negative value in 'has_angel' column : False
Is there any negative value in 'has_roundA' column : False
Is there any negative value in 'has_roundB' column : False
Is there any negative value in 'has_roundC' column : False
Is there any negative value in 'has_roundD' column : False
Is there any negative value in 'avg_participants' column : False
Is there any negative value in 'is_top500' column : False
```

```
df['status'] = df['status'].astype('category')
df['status'].replace(['acquired','closed'],[1, 0], inplace=True)
df['status']
```

```
0      1
1      1
2      1
3      1
4      0
      ..
918    1
919    0
920    0
921    1
922    1
Name: status, Length: 839, dtype: category
Categories (2, int64): [1, 0]
```

```
df['status'] = df['status'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 839 entries, 0 to 922
Data columns (total 47 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                839 non-null    int64
 1   state_code                839 non-null    object
 2   latitude                  839 non-null    float64
 3   longitude                 839 non-null    float64
 4   zip_code                  839 non-null    object
 5   id                        839 non-null    object
 6   city                      839 non-null    object
 7   name                      839 non-null    object
 8   labels                    839 non-null    int64
 9   founded_at                839 non-null    object
 10  first_funding_at          839 non-null    object
 11  last_funding_at           839 non-null    object
 12  age_first_funding_year    839 non-null    float64
 13  age_last_funding_year     839 non-null    float64
 14  age_first_milestone_year  839 non-null    float64
 15  age_last_milestone_year   839 non-null    float64
 16  relationships             839 non-null    int64
 17  funding_rounds            839 non-null    int64
 18  funding_total_usd         839 non-null    int64
 19  milestones                839 non-null    int64
 20  state_code.1              839 non-null    object
 21  is_CA                     839 non-null    int64
 22  is_NY                     839 non-null    int64
 23  is_MA                     839 non-null    int64
 24  is_TX                     839 non-null    int64
 25  is_otherstate             839 non-null    int64
 26  category_code             839 non-null    object
 27  is_software               839 non-null    int64
 28  is_web                    839 non-null    int64
 29  is_mobile                 839 non-null    int64
 30  is_enterprise             839 non-null    int64
 31  is_advertising            839 non-null    int64
 32  is_gamesvideo             839 non-null    int64
 33  is_ecommerce              839 non-null    int64
 34  is_biotech                839 non-null    int64
 35  is_consulting             839 non-null    int64
 36  is_othercategory          839 non-null    int64
 37  object_id                 839 non-null    object
 38  has_VC                    839 non-null    int64
 39  has_angel                 839 non-null    int64
 40  has_roundA                839 non-null    int64
 41  has_roundB                839 non-null    int64
 42  has_roundC                839 non-null    int64
 43  has_roundD                839 non-null    int64
 44  avg_participants          839 non-null    float64
 45  is_top500                 839 non-null    int64
 46  status                    839 non-null    int64
dtypes: float64(7), int64(29), object(11)
memory usage: 314.6+ KB
```

```
df.drop(['labels','state_code.1', ],axis=1, inplace=True)
```

```
df_cat1=df.select_dtypes(include='object')
df_cat1
```

	state_code	zip_code	id	city	name	founded_at	first_funding_
0	CA	92101	c:6669	San Diego	Bandsintown	1/1/2007	4/1/20
1	CA	95032	c:16283	Los Gatos	TriCipher	1/1/2000	2/14/20
2	CA	92121	c:65620	San Diego	Plixi	3/18/2009	3/30/20
3	CA	95014	c:42668	Cupertino	Solidcore Systems	1/1/2002	2/17/20
4	CA	94105	c:65806	San Francisco	Inhale Digital	8/1/2010	8/1/20
...	...	...	...	...	...	...	
918	CA	94107	c:21343	San Francisco	CoTweet	1/1/2009	7/9/20
919	MA	1803	c:41747	Burlington	Reef Point Systems	1/1/1998	4/1/20
920	CA	94089	c:31549	Sunnyvale	Paracor Medical	1/1/1999	6/29/20

```
df_cat1.nunique()
```

```
state_code            35
zip_code             359
id                   838
city                 210
name                 838
founded_at           189
first_funding_at     550
last_funding_at      640
category_code         35
object_id            838
dtype: int64
```

```
df.drop(['id','name','object_id' ],axis=1, inplace=True)
```

```
numeric1=['int8', 'int16', 'int32', 'int64', 'float16', 'float32', 'float64']
df_num1=df.select_dtypes(include=numeric1)
df_num1
```

	Unnamed: 0	latitude	longitude	age_first_funding_year	age_last_funding_year
0	1005	42.358880	-71.056820	2.2493	3.0027
1	204	37.238916	-121.973718	5.1260	9.9973
2	1001	32.901049	-117.192656	1.0329	1.0329
3	738	37.320309	-122.050040	3.1315	5.3151
4	1002	37.779281	-122.419236	0.0000	1.6685
...	...	...	...	...	...
918	352	37.740594	-122.376471	0.5178	0.5178
919	721	42.504817	-71.195611	7.2521	9.2274
920	557	37.408261	-122.015920	8.4959	8.4959
921	589	37.556732	-122.288378	0.7589	2.8329
922	462	37.386778	-121.966277	3.1205	3.1205

839 rows × 35 columns

```
df_num1.nunique()
```

```
Unnamed: 0                  839
latitude                    599
longitude                   598
age_first_funding_year      572
age_last_funding_year       697
age_first_milestone_year    410
age_last_milestone_year     522
relationships                38
funding_rounds                9
funding_total_usd           467
milestones                    8
is_CA                         2
is_NY                         2
is_MA                         2
is_TX                         2
is_otherstate                 2
is_software                   2
is_web                        2
is_mobile                     2
is_enterprise                 2
is_advertising                2
is_gamesvideo                 2
is_ecommerce                  2
is_biotech                    2
is_consulting                 2
is_othercategory              2
has_VC                        2
has_angel                     2
has_roundA                    2
has_roundB                    2
has_roundC                    2
has_roundD                    2
avg_participants             57
is_top500                     2
status                        2
dtype: int64
```

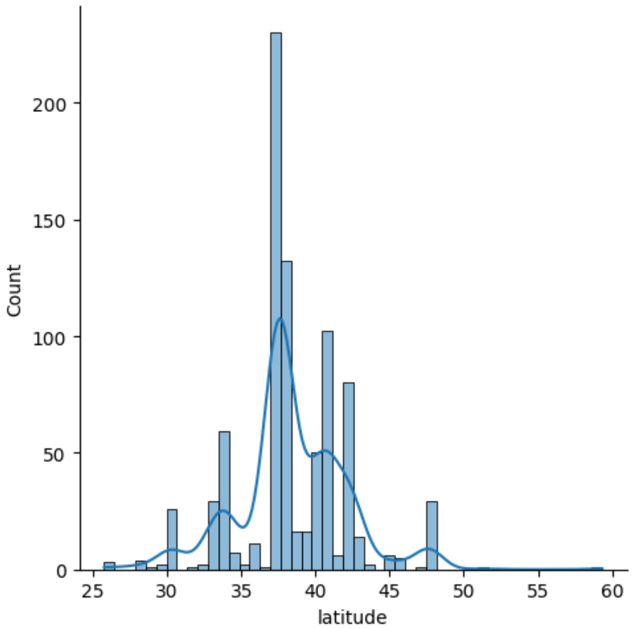
```
df.drop(['Unnamed: 0' ],axis=1, inplace=True)
```

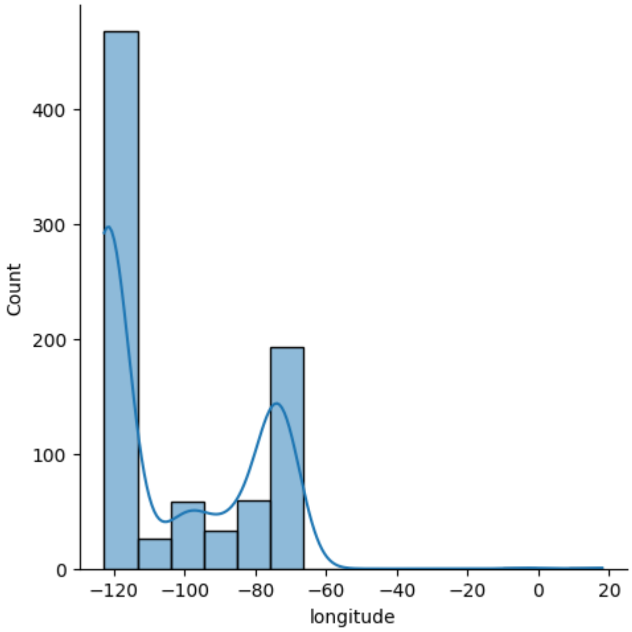
```
df.shape
```

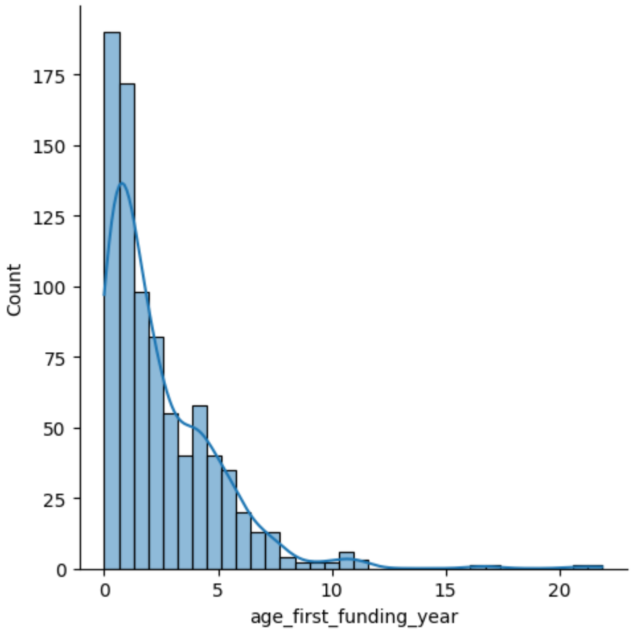
```
(839, 41)
```

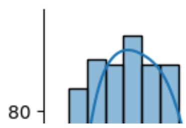
```
columns_outliers=['latitude', 'longitude', 'age_first_funding_year',
       'age_last_funding_year', 'age_first_milestone_year',
       'age_last_milestone_year', 'relationships', 'funding_rounds',
       'funding_total_usd', 'milestones', 'avg_participants']
```

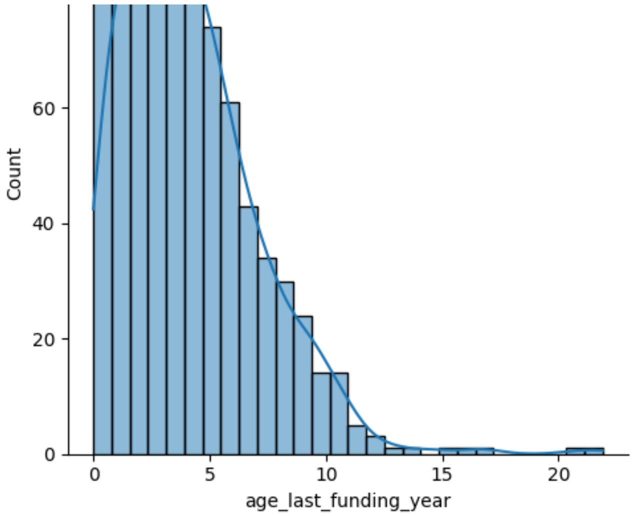
```
for i in columns_outliers:
  sns.displot(df[i], kde=True)
  plt.show()
```

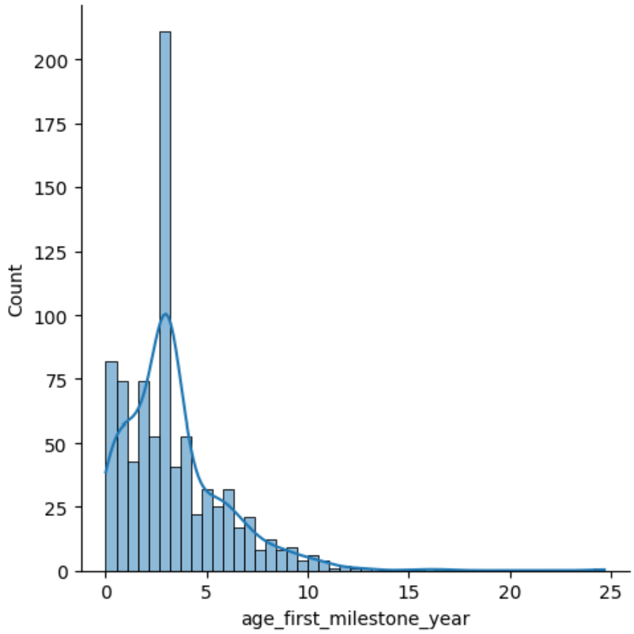

Count
0
50
100
150
200
25
30
35
40
45
50
55
60
latitude

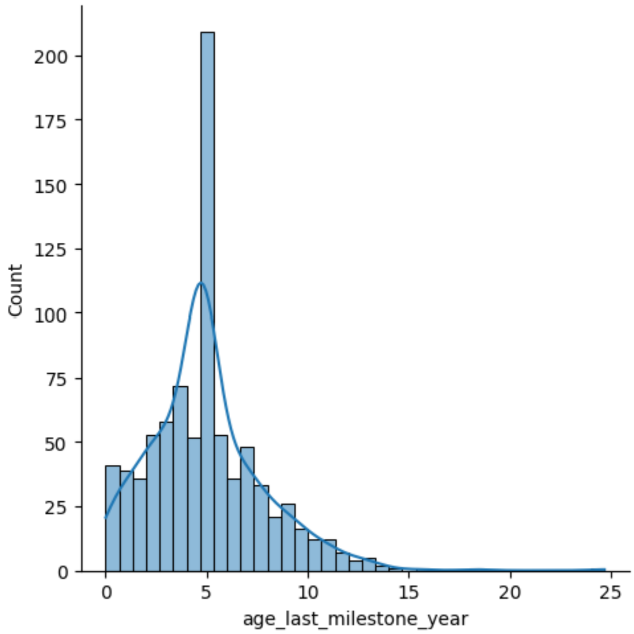

Count
0
100
200
300
400
−120
−100
−80
−60
−40
−20
0
20
longitude

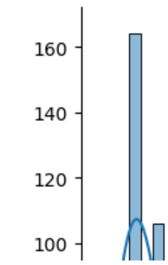

Count
0
25
50
75
100
125
150
175
0
5
10
15
20
age_first_funding_year

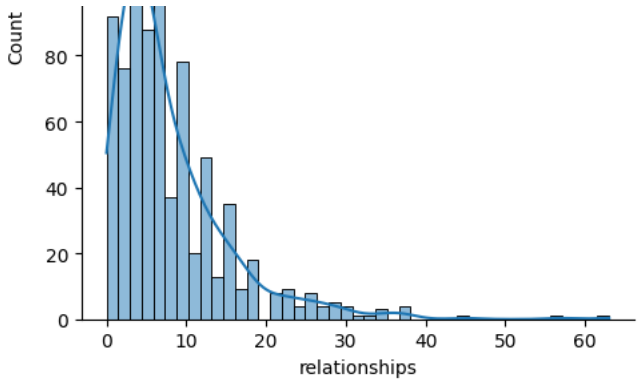

80

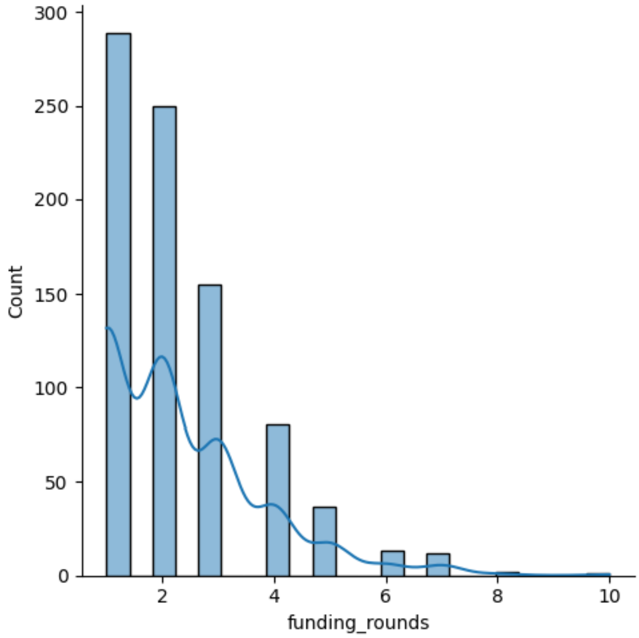

60
40
20
0
Count
0
5
10
15
20
age_last_funding_year

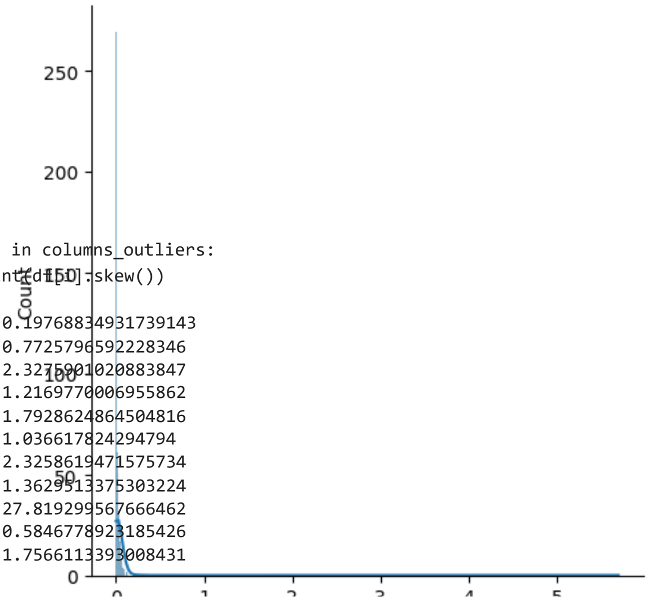

200
175
150
125
100
75
50
25
0
Count
0
5
10
15
20
25
age_first_milestone_year


200
175
150
125
100
75
50
25
0
Count
0
5
10
15
20
25
age_last_milestone_year


160
140
120
100



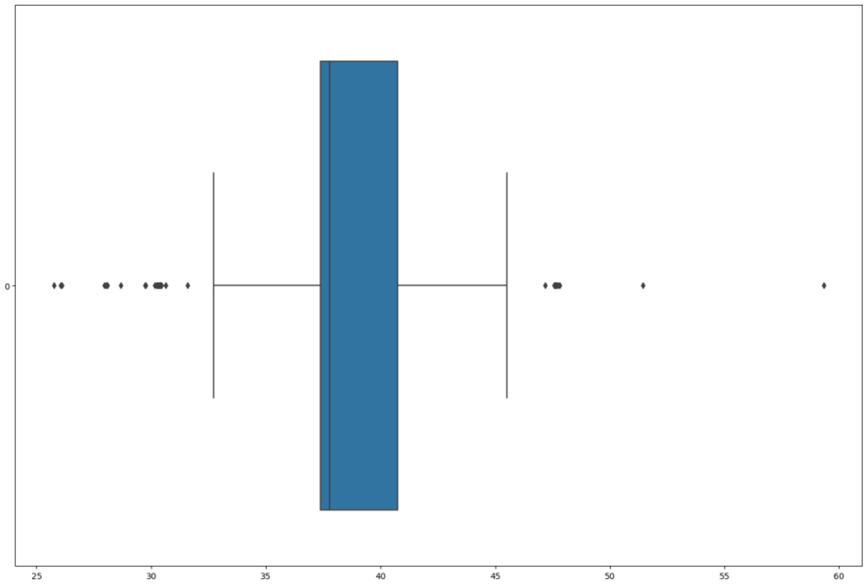


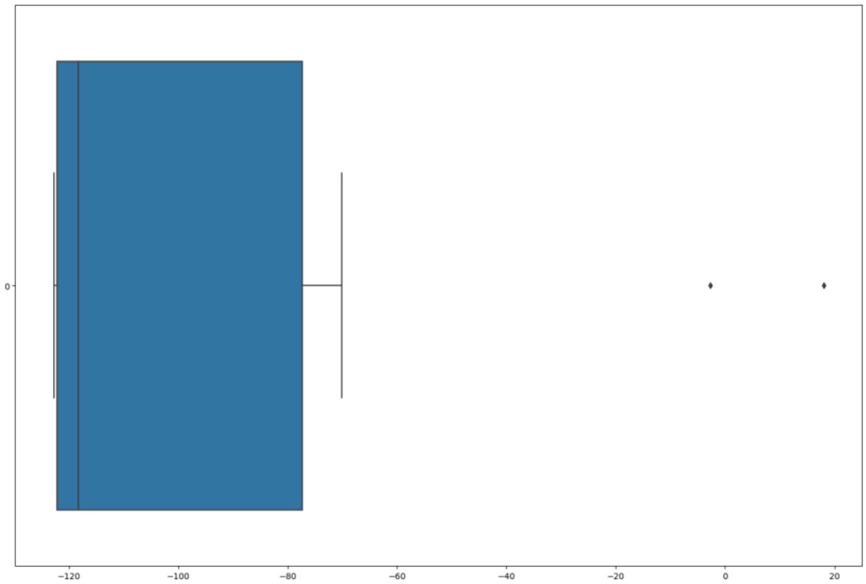


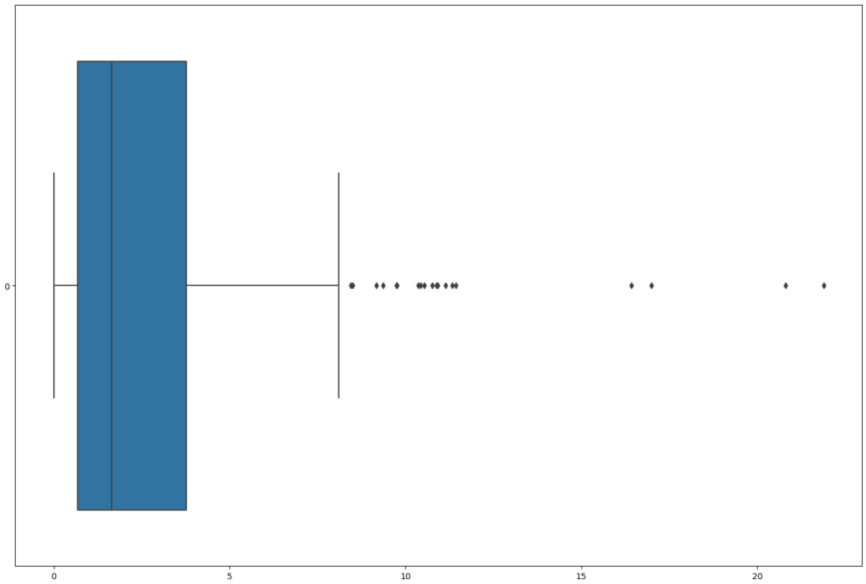

```
for i in columns_outliers:
  print(df[i].skew())
```

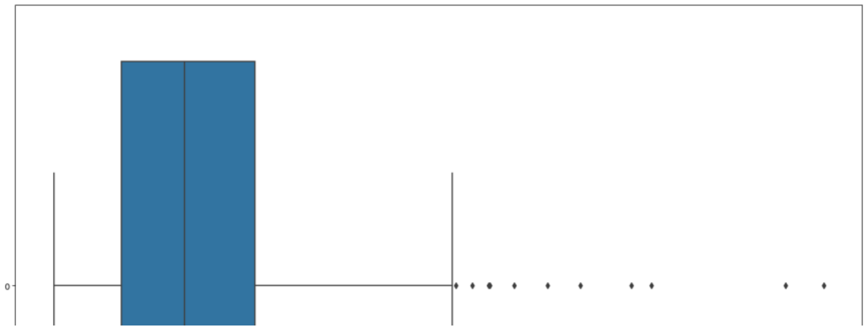
```
0.19768834931739143
0.7725796592228346
2.3275901020883847
1.2169770006955862
1.7928624864504816
1.036617824294794
2.3258619471575734
1.3629513375303224
27.819299567666462
0.5846778923185426
1.7566113393008431
```

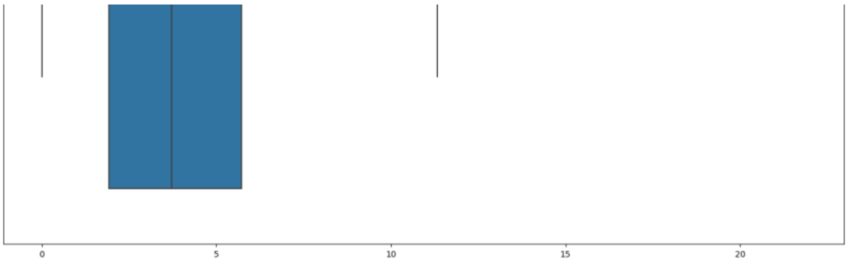
```
for i in columns_outliers:
  plt.figure(figsize=(18,12))
  sns.boxplot(df[i],orient='h')
  plt.show()
```

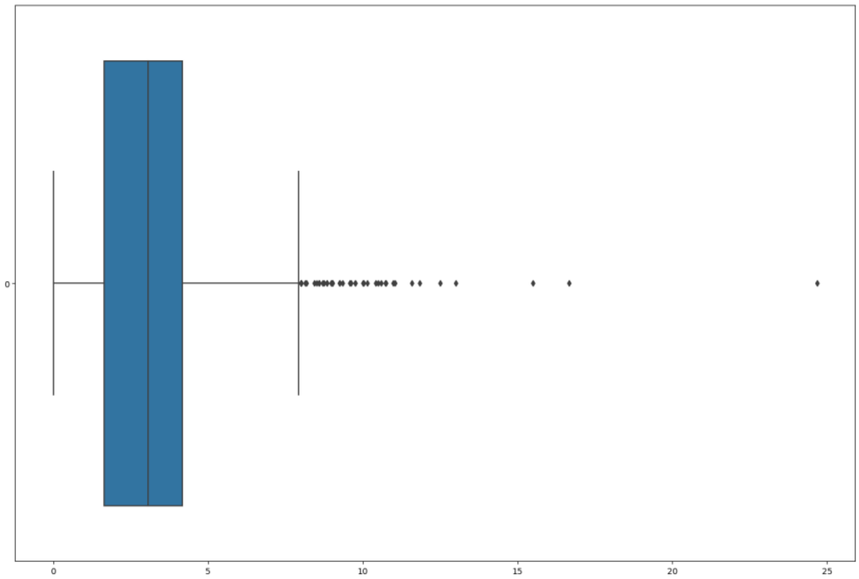

0
25
30
35
40
45
50
55
60

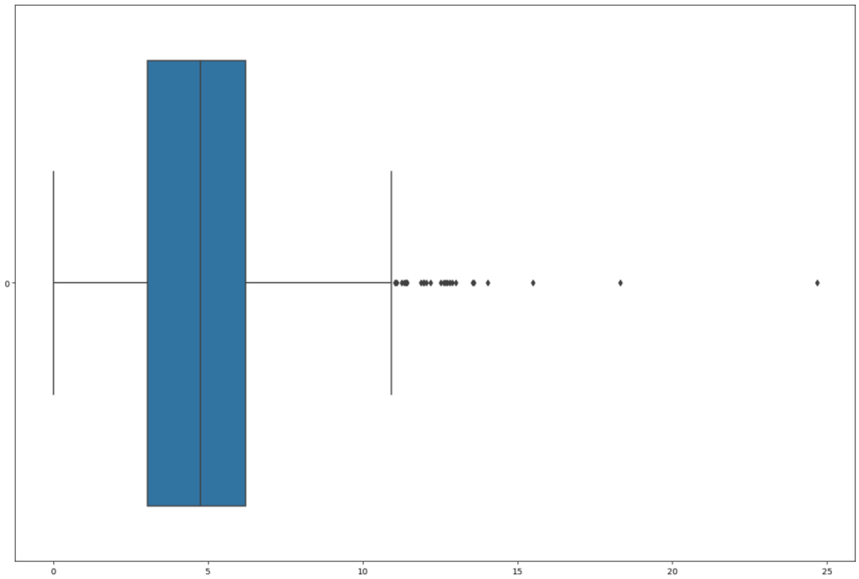

0
−120
−100
−80
−60
−40
−20
0
20

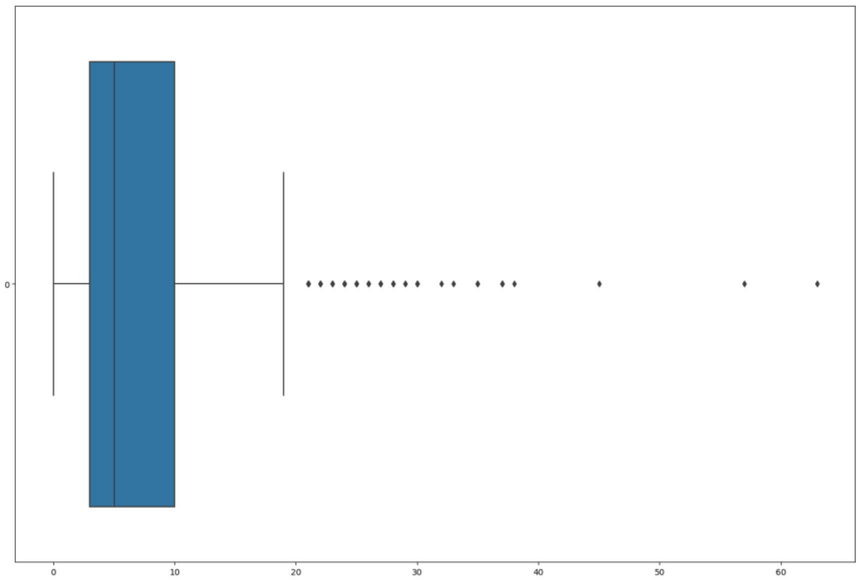

0
0
5
10
15
20


0


0
5
10
15
20


0
0
5
10
15
20
25


0
0
5
10
15
20
25


0
0
10
20
30
40
50
60

```
def remove_outliers(df, featuresNumfinal):
  for i in range(0, len(featuresNumfinal)):
   q1 =
   df[featuresNumfina
   l[i]].quantile(0.2
   5)    q3    =
   df[featuresNumfina
   l[i]].quantile(0.7
   5) iqr = q3 - q1
   lower_bound
   = q1 - 1.5 *
   iqr
   upper_bound
   = q3 + 1.5 *
   iqr
   cleaned_data          =
          df[(df[featuresNum
          final[i]]      >=
          lower_bound)    &
          (df[featuresNumfin
          al[i]]         <=
          upper_bound)]
   return cleaned_data

 featuresNumfinal=['latitude',
    'longitude',
    'age_first_funding_year',
    'age_last_funding_year',
    'age_first_milestone_year'
    ,
    'age_last_milestone_year',
```

```
    'relationships',
    'funding_rounds',
    'funding_total_usd',
    'milestones',
    'avg_participants']

new_df=remove_outl
iers(df,featuresNu
mfinal)
print(new_df.shape
)
```

```
(770, 41)
```

```
for i in
 columns_ou
 tliers:
 print(new_
 df[i].skew
 ())
```

```
-0.008486924516590088
0.5220350425662782
2.291553508040805
1.2139597965231217
1.8384458396165007
1.0882330363610442
2.33436886369956
1.3915021246926884
6.516656101840025
0.6059237389094657
1.7426382291702065
```

```
new_df
```

	state_code	latitude	longitude	zip_code	city	founded_at	first_funding
0	CA	42.358880	-71.056820	92101	San Diego	1/1/2007	4/1/2
1	CA	37.238916	-121.973718	95032	Los Gatos	1/1/2000	2/14/2
2	CA	32.901049	-117.192656	92121	San Diego	3/18/2009	3/30/2
3	CA	37.320309	-122.050040	95014	Cupertino	1/1/2002	2/17/2
4	CA	37.779281	-122.419236	94105	San Francisco	8/1/2010	8/1/2
...	...	...	...	...	...	...	
918	CA	37.740594	-122.376471	94107	San Francisco	1/1/2009	7/9/2
919	MA	42.504817	-71.195611	1803	Burlington	1/1/1998	4/1/2
920	CA	37.408261	-122.015920	94089	Sunnyvale	1/1/1999	6/29/2
921	CA	37.556732	-122.288378	94404	San Francisco	1/1/2009	10/5/2
922	CA	37.386778	-121.966277	95054	Santa Clara	1/1/2003	2/13/2

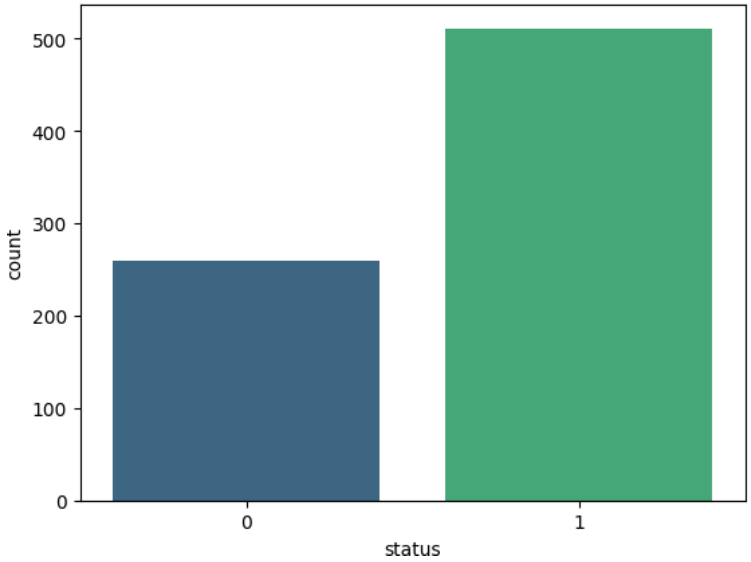
770 rows × 41 columns

```
new_df.drop(['founded_at','first_funding_at','last_funding_at' ],axis=1, inplace=True)
```

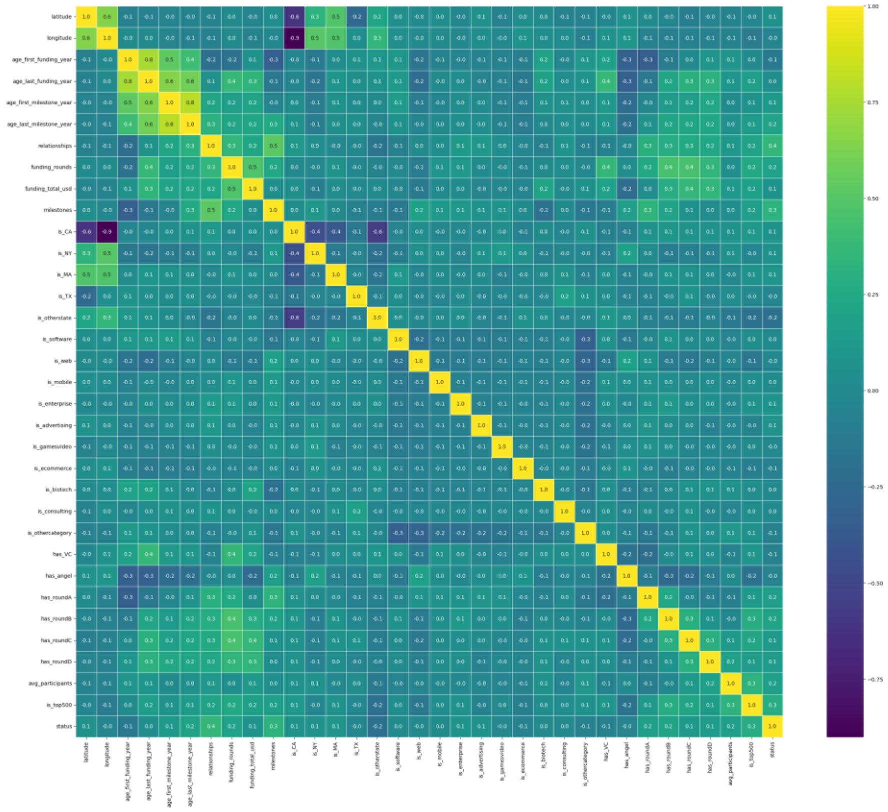
```
new_df.shape
```

```
(770, 38)
```

```
sns.countplot(x = new_df['status'], palette = 'viridis')
plt.show()
```

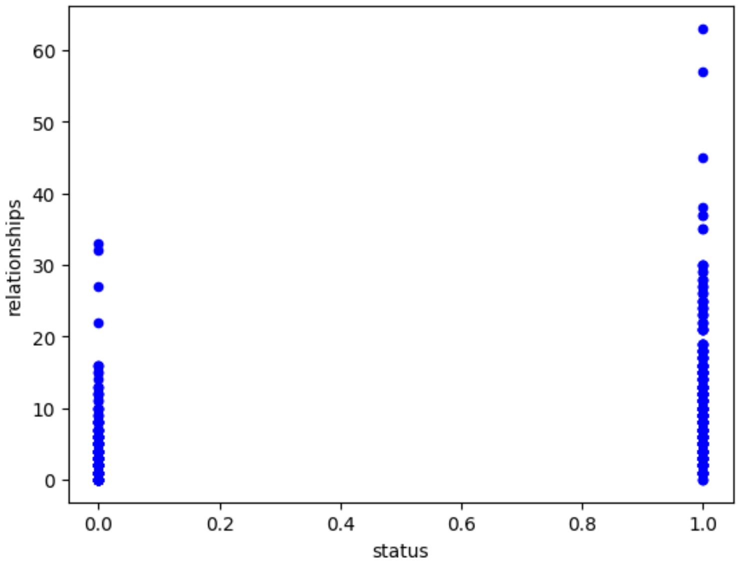



```
plt.figure(figsize = (30, 25))
sns.heatmap(new_df.corr(),annot = True, cmap = 'viridis', linewidth = 0.5, fmt = '.1f')
plt.show()
```



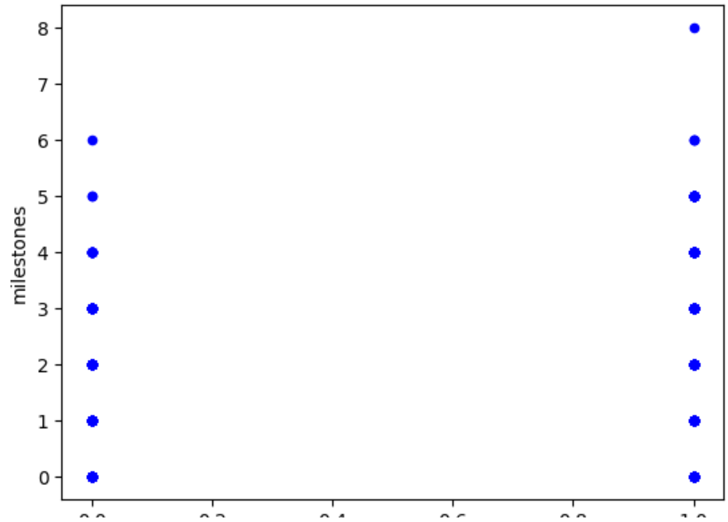
```
new_df.plot(kind='scatter',x='status',y='relationships',color='blue')
```

    <Axes: xlabel='status', ylabel='relationships'>



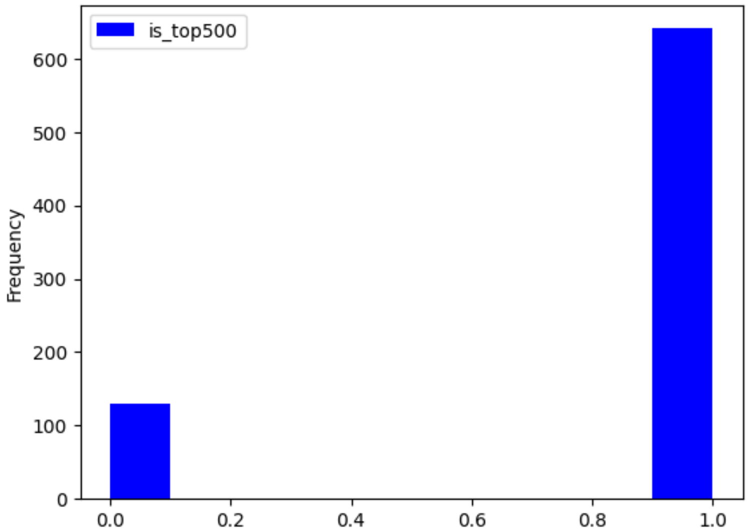

```
new_df.plot(kind='scatter',x='status',y='milestones',color='blue')
```

```
<Axes: xlabel='status', ylabel='milestones'>
```



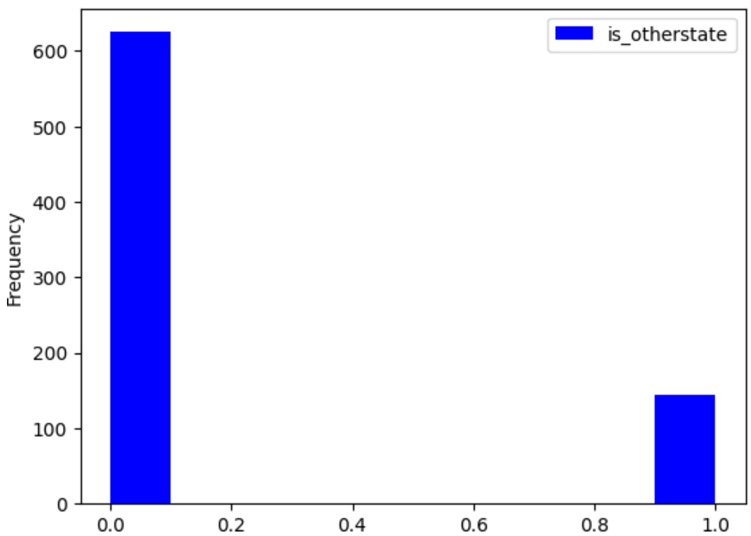

```
new_df.plot(kind='hist',x='status',y='is_top500',color='blue')
```

```
<Axes: ylabel='Frequency'>
```




```
new_df.plot(kind='hist',x='status',y='is_otherstate',color='blue')
```

```
<Axes: ylabel='Frequency'>
```




```
new_df.drop(['is_CA','is_NY',   'is_MA','is_TX','is_otherstate','is_software',
             'is_web','is_mobile','is_enterprise','is_advertising','is_gamesvideo',
             'is_ecommerce','is_biotech','is_consulting',
             'is_othercategory'],axis=1, inplace=True)
```

```
print(new_df.shape)
print(new_df.nunique())
```

```
print(new_df.info())
```

```
(770, 23)
state_code                  33
latitude                   542
longitude                  541
zip_code                   320
city                       195
age_first_funding_year     539
age_last_funding_year      653
age_first_milestone_year   385
age_last_milestone_year    485
relationships               38
funding_rounds               9
funding_total_usd          439
milestones                   8
category_code               35
has_VC                       2
has_angel                    2
has_roundA                   2
has_roundB                   2
has_roundC                   2
has_roundD                   2
avg_participants            53
is_top500                    2
status                       2
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Int64Index: 770 entries, 0 to 922
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   state_code                770 non-null    object
 1   latitude                  770 non-null    float64
 2   longitude                 770 non-null    float64
 3   zip_code                  770 non-null    object
 4   city                      770 non-null    object
 5   age_first_funding_year    770 non-null    float64
 6   age_last_funding_year     770 non-null    float64
 7   age_first_milestone_year  770 non-null    float64
 8   age_last_milestone_year   770 non-null    float64
 9   relationships             770 non-null    int64
 10  funding_rounds            770 non-null    int64
 11  funding_total_usd         770 non-null    int64
 12  milestones                770 non-null    int64
 13  category_code             770 non-null    object
 14  has_VC                    770 non-null    int64
 15  has_angel                 770 non-null    int64
 16  has_roundA                770 non-null    int64
 17  has_roundB                770 non-null    int64
 18  has_roundC                770 non-null    int64
 19  has_roundD                770 non-null    int64
 20  avg_participants          770 non-null    float64
 21  is_top500                 770 non-null    int64
 22  status                    770 non-null    int64
dtypes: float64(7), int64(12), object(4)
memory usage: 144.4+ KB
None
```

```
Le=LabelEncoder()
new_df['category_code']=Le.fit_transform(new_df['category_code'])
new_df['city']=Le.fit_transform(new_df['city'])
new_df['zip_code']=Le.fit_transform(new_df['zip_code'])
new_df['state_code']=Le.fit_transform(new_df['state_code'])
new_df[['state_code','zip_code','city','category_code']]
```

state_code	zip_code	city	category_code

```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 770 entries, 0 to 922
Data columns (total 23 columns):
 #    Column                    Non-Null Count  Dtype
----  ------                    --------------  -----
 0    state_code                770 non-null    int64
 1    latitude                  770 non-null    float64
 2    longitude                 770 non-null    float64
 3    zip_code                  770 non-null    int64
 4    city                      770 non-null    int64
 5    age_first_funding_year    770 non-null    float64
 6    age_last_funding_year     770 non-null    float64
 7    age_first_milestone_year  770 non-null    float64
 8    age_last_milestone_year   770 non-null    float64
 9    relationships             770 non-null    int64
 10   funding_rounds            770 non-null    int64
 11   funding_total_usd         770 non-null    int64
 12   milestones                770 non-null    int64
 13   category_code             770 non-null    int64
 14   has_VC                    770 non-null    int64
 15   has_angel                 770 non-null    int64
 16   has_roundA                770 non-null    int64
 17   has_roundB                770 non-null    int64
 18   has_roundC                770 non-null    int64
 19   has_roundD                770 non-null    int64
 20   avg_participants          770 non-null    float64
 21   is_top500                 770 non-null    int64
 22   status                    770 non-null    int64
dtypes: float64(7), int64(16)
memory usage: 144.4 KB
```

```
X=new_df.drop(['status'],axis=1)
y=new_df['status']
```

```
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.20,random_state=1)
```

```
x_train
```

	state_code	latitude	longitude	zip_code	city	age_first_funding_year	age_l
653	2	32.715400	-117.156500	215	154	0.3288	
681	22	40.757929	-73.985506	5	118	0.2466	
191	2	37.809338	-122.416606	262	155	0.7479	
505	2	37.563905	-122.324688	271	158	16.9863	
442	2	37.763652	-122.421778	261	155	0.1671	
...	...	...	...	...	...	...	
767	2	37.662431	-121.874679	281	141	3.1671	
862	19	40.707045	-74.956003	173	71	3.9726	
91	5	37.090240	-95.712891	62	184	1.1671	
282	2	37.779281	-122.419236	255	155	0.0000	
46	22	40.730646	-73.986614	4	125	0.9260	

616 rows × 22 columns

```
x_test
```

	state_code	latitude	longitude	zip_code	city	age_first_funding_year	age_l
338	22	40.756054	-73.986951	3	125	0.0000	
124	2	37.548270	-121.988572	278	67	0.7562	
914	22	40.750519	-73.993494	3	125	3.2137	
419	11	42.528635	-71.278022	47	20	4.3315	

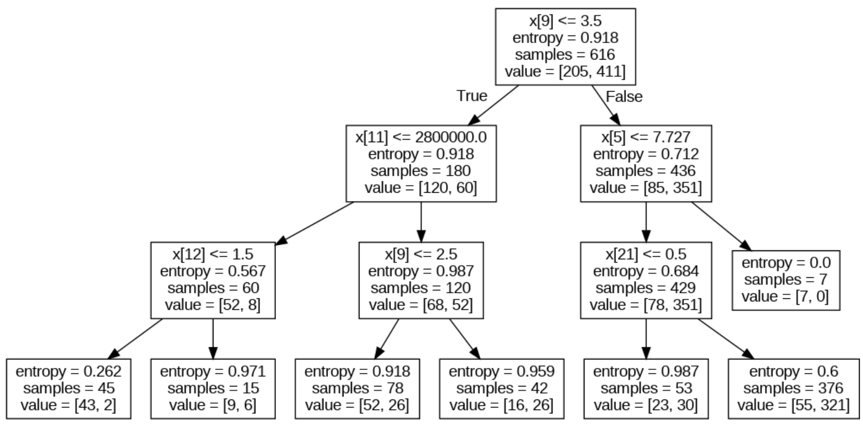
```
y_train
```

```
653    1
681    1
191    1
505    0
442    1
      ..
767    1
862    1
91     0
282    1
46     1
Name: status, Length: 616, dtype: int64
```

```
y_test
```

```
338    1
124    1
914    1
419    1
482    1
      ..
676    0
621    1
184    0
415    1
649    1
Name: status, Length: 154, dtype: int64
```

```
decision_tree = DecisionTreeClassifier(criterion='entropy',random_state=1,max_depth=3)
decision_tree = decision_tree.fit(x_train,y_train)
dot_data = export_graphviz(decision_tree, out_file=None)
graph = graphviz.Source(dot_data)
image = graph.render(format='png')
display(Image(image))
```

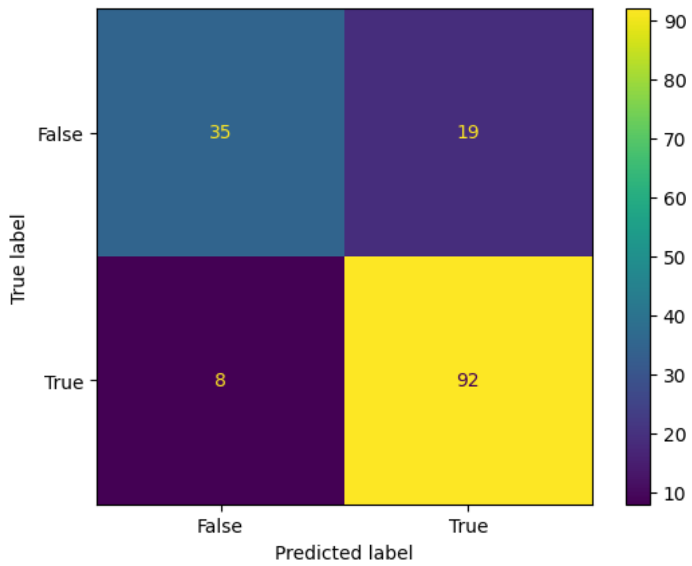



```
y_pred = decision_tree.predict(x_test)
y_test==y_pred
```

```
338    True
124    True
914    True
419    True
482    True
       ...
676    True
621    True
```

```
184     True
415     True
649     True
Name: status, Length: 154, dtype: bool
```

```
confusion_matrix = metrics.confusion_matrix(y_test,y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
plt.show()
```




```
accuracy = accuracy_score(y_test, y_pred)
print('Decision Tree Accuracy:', accuracy*100)
report = classification_report(y_test, y_pred)
print(' Decision Tree Classification report:\n', report)
```

```
Decision Tree Accuracy: 82.46753246753246
 Decision Tree Classification report:
               precision    recall  f1-score   support

           0       0.81      0.65      0.72        54
           1       0.83      0.92      0.87       100

    accuracy                           0.82       154
   macro avg       0.82      0.78      0.80       154
weighted avg       0.82      0.82      0.82       154
```

```
data = pd.DataFrame({
    'state_code':22,'latitude':40.756054, 'longitude':-73.986951,
    'zip_code':3,   'city':125, 'age_first_funding_year':0.0000,
    'age_last_funding_year':3.9562, 'age_first_milestone_year':4.7616,
    'age_last_milestone_year':5.1507,    'relationships':24, 'funding_rounds':5,
    'funding_total_usd':90000000,    'milestones':3, 'category_code':8,
    'has_VC':1, 'has_angel':0, 'has_roundA':1, 'has_roundB':1, 'has_roundC':1,
    'has_roundD':1, 'avg_participants':3.8000, 'is_top500':1
},index=[0])
data
```

	state_code	latitude	longitude	zip_code	city	age_first_funding_year	age_last
0	22	40.756054	-73.986951	3	125	0.0	

```
prediction=decision_tree.predict(data)
if prediction[0]==1:
  print("The prediction indicates the startup will be successful")
else:
  print("The prediction indicates the startup will not be successful")
```

```
The prediction indicates the startup will be successful
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score, mean_squared_error
```

```
random_forest = RandomForestClassifier(n_estimators=100, criterion='entropy', random_state=1, max_depth=3)
random_forest.fit(x_train, y_train)
```

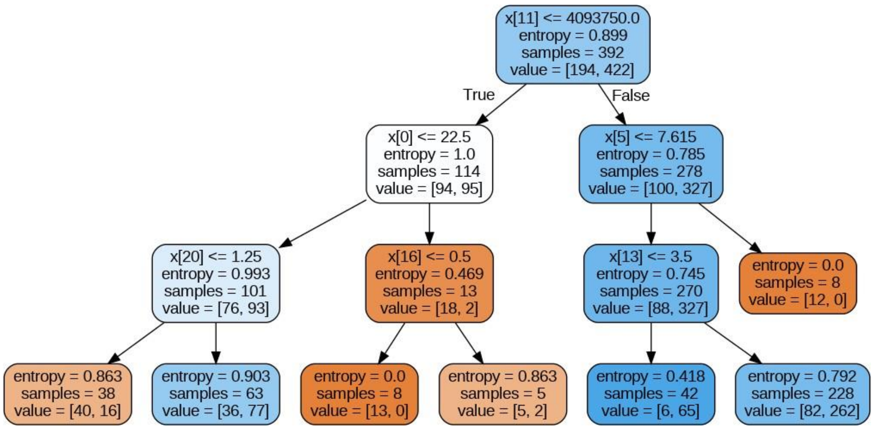
```
▾                          RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=3, random_state=1)
```

```
from sklearn.tree import export_graphviz
import graphviz
from IPython.display import Image, display
```

```
tree = random_forest.estimators_[0]

dot_data = export_graphviz(tree, out_file=None, filled=True, rounded=True)
graph = graphviz.Source(dot_data)
image = graph.render(format='png')

display(Image(image))
```




```
y_predr = random_forest.predict(x_test)
y_predr
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1])
```

```
confusion_matrixr = metrics.confusion_matrix(y_test,y_predr)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrixr, display_labels = [False, True])
cm_display.plot()
plt.show()
```


```
accuracyR = accuracy_score(y_test, y_predr)
accuracyR
```

```
0.7987012987012987
```

```
print('Random Forest Accuracy:', accuracyR*100)
report = classification_report(y_test, y_predr)
print(' Random Forest Classification report:\n', report)
```

```
Random Forest Accuracy: 79.87012987012987
 Random Forest Classification report:
               precision    recall  f1-score   support

           0       0.87      0.50      0.64        54
           1       0.78      0.96      0.86       100

    accuracy                           0.80       154
   macro avg       0.83      0.73      0.75       154
weighted avg       0.81      0.80      0.78       154
```

```
data = pd.DataFrame({
    'state_code':22,'latitude':40.756054, 'longitude':-73.986951,
    'zip_code':3,   'city':125, 'age_first_funding_year':0.0000,
    'age_last_funding_year':3.9562, 'age_first_milestone_year':4.7616,
    'age_last_milestone_year':5.1507,   'relationships':24, 'funding_rounds':5,
    'funding_total_usd':90000000,   'milestones':3, 'category_code':8,
    'has_VC':1, 'has_angel':0, 'has_roundA':1, 'has_roundB':1, 'has_roundC':1,
    'has_roundD':1, 'avg_participants':3.8000, 'is_top500':1
},index=[0])
data
```

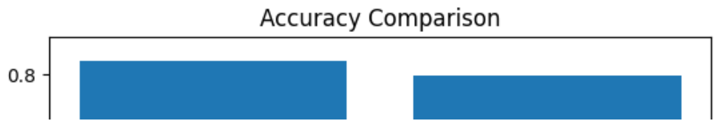
	state_code	latitude	longitude	zip_code	city	age_first_funding_year	age_last
0	22	40.756054	-73.986951	3	125	0.0	

```
prediction=random_forest.predict(data)
if prediction[0]==1:
  print("The prediction indicates the startup will be successful")
else:
  print("The prediction indicates the startup will not be successful")
```

```
The prediction indicates the startup will be successful
```

```
accuracy_scores = [accuracy, accuracyR]
algorithms = ["Decicion Tree Classifier", "Random Forest Classifiers"]

plt.bar(algorithms, accuracy_scores)
plt.xlabel("Algorithms")
plt.ylabel("Accuracy")
plt.title("Accuracy Comparison")
plt.show()
```



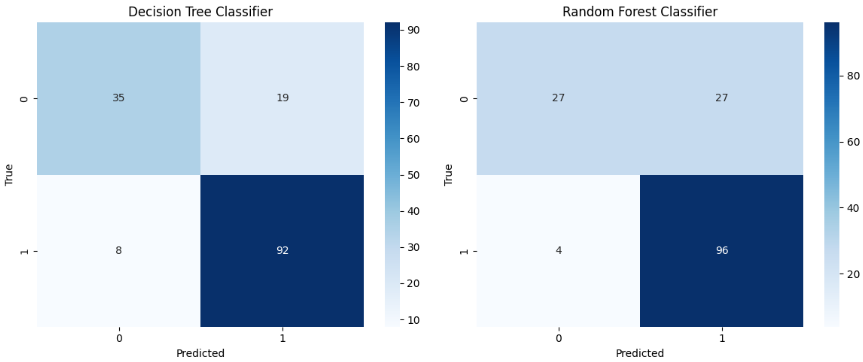

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

sns.heatmap(confusion_matrix, annot=True, fmt="d", cmap="Blues", ax=axes[0])
axes[0].set_title("Decision Tree Classifier")
axes[0].set_xlabel("Predicted")
axes[0].set_ylabel("True")

sns.heatmap(confusion_matrixr, annot=True, fmt="d", cmap="Blues", ax=axes[1])
axes[1].set_title("Random Forest Classifier")
axes[1].set_xlabel("Predicted")
axes[1].set_ylabel("True")

plt.tight_layout()
plt.show()
```




```
import pickle


with open("decisiontree.pkl", "wb") as f:
  pickle.dump(decision_tree, f)


with open("decisiontree.pkl", "rb") as f:
    clf = pickle.load(f)


clf
```

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=1)
```

```
def make_prediction(state_code,latitude,longitude,zip_code,city,age_first_funding_year,age_last_funding_year,age_first_milestone_year,age
                              ,funding_rounds,funding_total_usd,milestones,category_code,has_VC
                              ,has_angel,has_roundA,has_roundB,has_roundC,has_roundD,avg_participants,is_top500):
    with open("decisiontree.pkl", "rb") as f:
        clf  = pickle.load(f)
        preds = clf.predict([[state_code,latitude,longitude,zip_code,city,age_first_funding_year,age_last_funding_year,age_first_mileston
                              ,funding_rounds,funding_total_usd,milestones,category_code,has_VC
                              ,has_angel,has_roundA,has_roundB,has_roundC,has_roundD,avg_participants,is_top500]])


        if preds == 1:
```

```
          return "Your startup is likely to be successful..."
        return "Your startup is not likely to be successful"



#Create the input component for Gradio since we are expecting 4 inputs

state_code=gr.Number(label="Enter state code")
latitude=gr.Number(label="Enter latitude")
longitude=gr.Number(label="Enter longitude")
zip_code=gr.Number(label="Enter zip_code")
city=gr.Number(label="Enter city")
age_first_funding_year=gr.Number(label="Enter age_first_funding_year")
age_last_funding_year=gr.Number(label="Enter age_last_funding_year")
age_first_milestone_year=gr.Number(label="Enter age_first_milestone_year")
age_last_milestone_year=gr.Number(label="Enter age_last_milestone_year")
relationships=gr.Number(label="Enter relationships")
funding_rounds=gr.Number(label="Enter funding_rounds")

funding_total_usd=gr.Number(label="Enter funding_total_usd")
milestones=gr.Number(label="Enter milestones")
category_code=gr.Number(label="Enter category_code")
has_VC=gr.Number(label="Enter has_VC")
has_angel=gr.Number(label="Enter has_angel")
has_roundA=gr.Number(label="Enter has_roundA")
has_roundB=gr.Number(label="Enter has_roundB")
has_roundC=gr.Number(label="Enter has_roundC")
has_roundD=gr.Number(label="Enter has_roundD")
avg_participants=gr.Number(label="Enter avg_participants")
is_top500=gr.Number(label="Enter is_top500")


# We create the output
output = gr.Textbox()


app = gr.Interface(fn = make_prediction, inputs=[state_code,latitude,longitude,zip_code,city,age_first_funding_year,age_last_funding_year
                             ,funding_rounds,funding_total_usd,milestones,category_code,has_VC
                             ,has_angel,has_roundA,has_roundB,has_roundC,has_roundD,avg_participants,is_top500], outputs=output)
app.launch()
```

```
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True`

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://cd31e4cf6067c56b5a.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run
```

Enter has_roundC

0

Enter has_roundD

0

Enter avg_participants

0

Enter is_top500

0

Clear **Submit**

Results:

- The Decision Tree algorithm achieved an accuracy of 82% in predicting startupsuccess.
- The Random Forest algorithm achieved an accuracy of 79% in predicting startupsuccess.
- The frontend provided an intuitive way for users to input data and obtain predictionsfrom the models.

Conclusion:

In this distributed computing project, we successfully preprocessed a dataset of startups across the USA, compared the performance of Decision Tree and Random Forest algorithms,and developed a user-friendly frontend for making predictions based on the machine learningmodels.

The project highlights the significance of distributed computing in data analysis and machinelearning and demonstrates the potential for applying these techniques to real-world business scenarios, such as predicting the success of startups.

Future Vision:

This project can be extended in various domains and achieve more accurate and wider range of predictions

1. **Geographical Location:** The model can be further linked with GPS Navigation to pin point better performing geographical location for a business outlet.

2. **Customer Behavior Analysis:** This model can help analyze customer behavior among other factors to improve marketing techniques.

3. **Risk Management:** Business expansion can be done with lesser risk accounting for many real time factors.

4. **Real-Time Monitoring:** Business Chains can better operate by monitoring all the data in real time from various branches after applying mining algorithms.

References:

1. Success prediction Dataset, Manish KC Momo- https://www.kaggle.com/datasets/manishkc06/startup-success-prediction
2. Deploying our model - https://www.freecodecamp.org/news/how-to-deploy-your-machine-learning-model-as-a-web-app-using-gradio/
3. Project Documentation format - https://www.cs.utexas.edu/ (University of Texas)