INDIAN STARTUPS

AMSTERDAM

Oh-so-famous central canal, rightly dubbed a UNESCO World Heritage Site in 2010. Add to that swathe of green spaces, storied red-brick facades, and museums filled with Van Gogh paintings, and you have yourself one of Europe's most gorgeous culture epicenters

PROJECT TITLE

Understanding the Indian Start-up Ecosystem: Analysis and Recommendations

INTRODUCTION

India has one of the fastest expanding economies in the world. Startups may be small businesses, but they can have a huge impact on economic growth. They generate more jobs, which leads to a healthier economy. Not only that, but startups can also contribute to economic vitality by encouraging innovation and injecting competition.

This project aims to provide insights into the Indian start-up ecosystem, including its current state, trends, and potential opportunities for growth. To achieve this, we will be analyzing key metrics in funding received by startups in India from 2018 to 2021. Through data analysis and visualization, we aim to identify key areas of focus for our team to enter the market and make a significant impact.

QUESTION

1. Which type of start-up location gets the most funding?
2. At which stage do start-ups get more funding from investors?
3. Which type of investors invest the most money?
4. Which type of investors invested the least money?
5. What is the percentage of Technology and Non-Technology in the Indian Start-up?

HYPOTHESIS

The method of hypothesis testing involves performing statistical tests on a sample to generate inferences or conclusions about the whole dataset.

To further analyze the data, we developed the null and alternate hypothesis to focus on two groups; Technology-biased startups and non-tech biased startups. So, the two hypotheses are as follows;

NULL: Technological industries do not have a higher success rate of being funded

ALTERNATE: Technological industries have a higher success rate of being funded

DATA UNDERSTANDING

This dataset has funding information of the Indian startups from 2018 to August 2021. It includes columns with the date funded, the city the startup is based, the names of the funders, and the amount invested.

In this phase, we aim at having a good understanding of the dataset in order to ascertain fact and some assumptions that will us to make informed decisions.

Data understanding is a critical step in the data science process, where you explore, visualize and get a deeper understanding of your dataset. In Python, there are many libraries and tools available that can help you with data understanding, such as NumPy, Pandas, Matplotlib, Seaborn, and more.

Here are some steps you can follow to perform data understanding in Python:

1. Import the necessary libraries: NumPy, Pandas, Matplotlib, and Seaborn are some of the commonly used libraries for data understanding.

    a. Pandas – pandas is a popular library for data manipulation and analysis. It provides data structures for efficient handling of structured data, including tables, series and data frames
    b. Numpy - Numpy is a library for numerical computing in python. It provides powerful tools for performing mathematical operations on large arrays and matrices.
    c. Matplotlib – is a data visualization library in python. It provides a range of plots and charts for exploring data visually, including scatter plots, bar charts, line charts and more
    d. Scikit-learn is also a popular machine learning library in python. It provides tools for data preprocessing, feature selections, model training, and evaluation.
    e. Seaborn is a data visualization library built on top of matplotlib. It provides a high-level interface for creating more sophisticated and visually appealing statistical graphics

    To start exploring the data, we imported the following libraries to help us view our data. We imported these libraries using the following commands.

Library importation

In [5]:

```
# Data handling
import numpy as np
import pandas as pd
import glob

# Vizualisation (Matplotlib, Plotly, Seaborn, etc. )
import matplotlib.ticker as ticker
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('whitegrid')

# import plotly.express as px


from scipy import stats

from scipy.stats import pearsonr

from scipy.stats import chi2_contingency
import plotly.express as px

import seaborn as sns
import matplotlib.pyplot as plt

# Other packages
import os
import re
#display all columns and rows
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
from sklearn.impute import SimpleImputer


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

Fig 1.0: Library importation

After importing the necessary libraries, we used various pandas functions to explore and analyze the data. For example, .head(), .describe(), function to view the first few rows of the data.

First, we imported the 2018 dataset with some specific columns that are relevant in our analysis.

Dataset Importation

```
# importing 2018 data separate it has different form from the other ones
startup2018=pd.read_csv('startup_funding2018.csv',
                        usecols = ['Company Name', 'Industry', 'Round/Series', 'Amount','Location'])

```

Fig 1.1: 2018 dataset importation

```
startup2018.head()
```

	Company Name	Industry	Round/Series	Amount	Location
0	TheCollegeFever	Brand Marketing, Event Promotion, Marketing, S...	Seed	250000	Bangalore, Karnataka, India
1	Happy Cow Dairy	Agriculture, Farming	Seed	₹40,000,000	Mumbai, Maharashtra, India
2	MyLoanCare	Credit, Financial Services, Lending, Marketplace	Series A	₹65,000,000	Gurgaon, Haryana, India
3	PayMe India	Financial Services, FinTech	Angel	2000000	Noida, Uttar Pradesh, India
4	Eunimart	E-Commerce Platforms, Retail, SaaS	Seed	—	Hyderabad, Andhra Pradesh, India

Fig 1.2: first five rows of the 2018 dataset

Taking a deep look at the dataset across the years, certain columns were common in all the datasets and were deemed important in the analysis. So, in importing the datasets, specific columns were imported. These included Company Name, Industry, Round/Series, Amount and Location

To ensure consistency, some columns were renamed to match the columns in the dataset of subsequent years. Industry was renamed to Sector and Round/Series to Stage

A new column named Funding Year was also added to each dataset and all rows were assigned the value 2018, depending on the dataset.

With the exception of 2018, all the other datasets had other relevant columns such as the year the startup was founded as well as the investor that provided funding. These columns were imported because there was a need to understand if the number of years that a startup had been.

```
# renaming columns for consistency
# Industry = sector
# Round/Series = stage
startup2018.rename(columns ={'Industry':'Sector'}, inplace = True)
startup2018.rename(columns ={'Round/Series':'Stage'}, inplace = True)
startup2018.rename(columns ={'Amount':'Amount($)'}, inplace = True)
# adding new column funding year
startup2018['Funding Year']= "2018"

# Changing the founding year into integer
startup2018['Funding Year'] =startup2018['Funding Year'].astype(int)
```

Fig 1.3: renaming of some columns

We realize that the amount column is in object data type that we needed to change to numeric datatype in order to enable us to convert the currency from Rupees to dollar to ensure consistency. We used the standard average exchange within the year in review.

It worth to note that some figures in the amount columns had no currency symbol attached to them, we assumed those figures to be in dollars.

We implemented this in the following steps

```
# getting the index all rows in the column amount that has rupees
get_index=startup2018.index[startup2018['Amount($)'].str.contains('₹')]

# charging the rows in rupees to dollars using standard rate

startup2018.loc[get_index,['Amount($)']]=startup2018.loc[get_index,['Amount($)']].values*0.012

startup2018.loc[:,['Amount($)']].head()

```

Fig. 1.4: converting the Rupees to dollars.

```
# Removing the symbols and commas from the amount column ₹, $, and ,
startup2018['Amount($)'] = startup2018['Amount($)'].apply(lambda x: str(x).replace('₹', ''))
startup2018['Amount($)'] = startup2018['Amount($)'].apply(lambda x: str(x).replace('$', ''))
startup2018['Amount($)'] = startup2018['Amount($)'].apply(lambda x: str(x).replace(',', ''))
```

Fig. 1.5: Removing symbol, comas in the amount column.

Just like we started in the opening statement, the 2018 dataset was in a different format so we decided to clear it first to ensure that it is in consistency with the 2019, 2020, and 2021 dataset.

From this phase we import and concatenated all the four datasets into our notebook as seen below

We use the glob function to list all our datasets in the specific folder.

In [7]:
```
csv_files = glob.glob('*.{}'.format('csv'))
csv_files
```

Out[7]:
```
['cleaned_2018.csv',
 'cleaned_2019.csv',
 'cleaned_2020.csv',
 'cleaned_2021.csv']
```

In [8]:
```
#Reading the data as DataFrame
data_final=pd.DataFrame()

for file in csv_files:
    df=pd.read_csv(file)

    data_final=data_final.append(df, ignore_index=True)
```

Then we used the '.column' to display all the column headings in order to trim the data to only the one needed for the analysis

```
data_final.columns
```

```
Index(['Unnamed: 0.1', 'Unnamed: 0', 'Company Name', 'Sector', 'Stage',
       'Amount($)', 'Location', 'Funding Year', 'What it does', 'Founders',
       'Investor', 'Unnamed: 9'],
      dtype='object')
```

```
data_final.drop(['Unnamed: 9'], axis=1, inplace=True)
data_final.drop(['Unnamed: 0'], axis=1, inplace=True)
data_final.drop(['What it does'], axis=1, inplace=True)
data_final.drop(['Unnamed: 0.1'], axis=1, inplace=True)
```

```
data_final.columns
```

```
Index(['Company Name', 'Sector', 'Stage', 'Amount($)', 'Location',
       'Funding Year', 'Founders', 'Investor'],
      dtype='object')
```

We dropped Unnamed: 9, Unnamed: 0, what it does and finally Unnamed: 0.1 since those columns are not necessary in our analysis.

```
1 data_final.head()
```

	Company Name	Sector	Stage	Amount($)	Location	Funding Year	Founders	Investor
0	TheCollegeFever	Brand Marketing	Seed	250000.0	Bangalore	2018.0	NaN	NaN
1	PayMe India	Financial Services	Angel	2000000.0	Noida	2018.0	NaN	NaN
2	Eunimart	E-Commerce Platforms	Seed	NaN	Hyderabad	2018.0	NaN	NaN
3	Hasura	Cloud Infrastructure	Seed	1600000.0	Bengaluru	2018.0	NaN	NaN
4	Freightwalla	Information Services	Seed	NaN	Mumbai	2018.0	NaN	NaN

Then we used the .head() method to display the top 5 rows of the dataset.

After getting all the required columns heads, what we needed to check again is the data types of those columns. We used the .info method to do that

```
1 data_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2734 entries, 0 to 2733
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Company Name  2734 non-null   object
 1   Sector        2716 non-null   object
 2   Stage         1796 non-null   object
 3   Amount($)     2580 non-null   object
 4   Location      2620 non-null   object
 5   Funding Year  2492 non-null   object
 6   Founders      2334 non-null   object
 7   Investor      2253 non-null   object
dtypes: object(8)
memory usage: 171.0+ KB
```

We detected that, the Amount and Funding Year columns had their data types as object (strings). We change them to float and data respectively.

```
1 data_final['Amount($)']=pd.to_numeric(data_final['Amount($)'], errors='coerce')
```

```
1 data_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2734 entries, 0 to 2733
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Company Name  2734 non-null   object
 1   Sector        2716 non-null   object
 2   Stage         1796 non-null   object
 3   Amount($)     233 non-null    float64
 4   Location      2620 non-null   object
 5   Funding Year  2492 non-null   object
 6   Founders      2334 non-null   object
 7   Investor      2253 non-null   object
dtypes: float64(1), object(7)
memory usage: 171.0+ KB
```

What we did again was to use to the .unique method to display all the unique items in the amount column so we can work on them.

```
3]: 1 data_final['Amount($)'].unique()
```

```
3]: array([2.50000e+05, 2.00000e+06,         nan, 1.60000e+06, 1.50000e+05,
           1.10000e+06, 6.00000e+06, 6.50000e+05, 1.00000e+06, 5.00000e+06,
           4.00000e+06, 2.80000e+06, 1.70000e+06, 1.30000e+06, 5.00000e+05,
           1.34000e+07, 9.00000e+06, 1.00000e+05, 2.00000e+04, 1.20000e+05,
           1.43145e+05, 7.42000e+08, 3.98000e+06, 1.00000e+04, 1.00000e+09,
           7.00000e+06, 3.50000e+07, 2.85000e+07, 2.40000e+06, 3.00000e+07,
           2.30000e+07, 1.10000e+07, 3.24000e+06, 5.40000e+08, 9.00000e+05,
           1.00000e+07, 1.50000e+06, 1.40000e+07, 1.00000e+08, 8.00000e+05,
           1.04100e+06, 1.50000e+04, 1.40000e+06, 1.20000e+06, 2.20000e+06,
           1.80000e+06, 3.60000e+06, 3.00000e+05, 6.83000e+06, 2.00000e+05,
           4.30000e+06, 3.64846e+05, 4.00000e+05, 1.32000e+07, 5.00000e+04,
           3.00000e+06, 1.25000e+06, 1.80000e+05, 4.20000e+06, 1.75000e+05,
           1.45000e+06, 4.50000e+06, 6.00000e+05, 1.50000e+07, 1.25000e+05,
           1.30000e+05, 1.72000e+07, 3.50000e+06, 1.20000e+07, 4.00000e+07,
           5.00000e+07, 4.19000e+07, 3.53000e+06, 3.30000e+06, 2.10000e+08,
           3.76800e+07, 2.20000e+07, 7.00000e+04, 1.85000e+08, 6.50000e+07,
           7.00000e+05, 7.50000e+07, 1.76000e+06, 2.70000e+06, 7.50000e+05,
           2.50000e+06, 8.00000e+07, 2.50000e+07, 3.70000e+06, 5.60000e+06,
           9.92300e+07, 7.00000e+07, 4.00000e+04, 5.50000e+05, 3.65000e+08,
           2.80000e+07, 1.49000e+07, 2.25000e+08, 7.50000e+03])
```

We also applied the unique method on the funding Year column to see the full unique items in the that column. We then replaced all NaNs and '-'. We replaced the NaNs and '-'with an empty string.

```
data_final['Funding Year'].unique()
```

```
array([2018.0, nan, 2014.0, 2004.0, 2013.0, 2010.0, 2019.0, 2017.0,
       2011.0, 2015.0, 2016.0, 2012.0, 2008.0, '2019', '2018', '2020',
       '2016', '2008', '2015', '2017', '2014', '1998', '2007', '2011',
       '1982', '2013', '2009', '2012', '1995', '2010', '2006', '1978',
       '1999', '1994', '2005', '1973', '-', '2002', '2004', '2001',
       2021.0, 2020.0, 1993.0, 1999.0, 1989.0, 2009.0, 2002.0, 1994.0,
       2006.0, 2000.0, 2007.0, 1978.0, 2003.0, 1998.0, 1991.0, 1984.0,
       2005.0, 1963.0], dtype=object)
```

```
data_final['Funding Year'] = data_final['Funding Year'].apply(lambda x: str(x).replace('-', ''))
```

```
data_final['Funding Year'] = data_final['Funding Year'].apply(lambda x: str(x).replace('nan', ''))
```

Finally, we applied the simple imputed method to replace the empty strings with the mean of Funding Year.

```
imp=SimpleImputer(strategy='mean')
data_final['Funding Year']=imp.fit_transform(data_final['Funding Year'].values.reshape(-1,1))
data_final['Funding Year'].isna().sum()
```

What we did again was to use the isna() method to check the percentage of all missing values. The code below helped us to achieve this.

```
missing_percentage=data_final.isna().mean()*100
```

```
missing_percentage
```

```
Company Name     0.000000
Sector           0.658376
Stage           34.308705
Amount($)       91.477688
Location         4.169715
Funding Year     0.000000
Founders        14.630578
Investor        17.593270
dtype: float64
```

We realized that Company Name and Funding Year had no missing values. Our attention is turn to the other columns

At the ‘stage’ column, we used the .unique() method to display the unique items in that field, it was revealed that one of the rows had url as a stage name which we removed. It is seen below

```
data_final['Stage'].unique()
```

```
array(['Seed', 'Angel', 'Series A', 'Pre-Seed', 'Private Equity',
       'Series B', 'Grant', 'Series H', 'Series C',
       'Venture - Series Unknown', 'Debt Financing', 'Post-IPO Debt',
       'Series E', 'Corporate Round',
       'https://docs.google.com/spreadsheets/d/1x9ziNeaz6auNChIHnMI8U6kS7knTr3byy_YBGfQaoUA/edit#gid=1861303593',
       'Series D', 'Secondary Market', 'Post-IPO Equity',
       'Non-equity Assistance', 'Funding Round', nan, 'Fresh funding',
       'Pre series A', 'Series G', 'Post series A', 'Seed funding',
       'Seed fund', 'Series F', 'Series B+', 'Seed round', 'Pre-series A',
       'Pre-seed', 'Pre-series', 'Debt', 'Pre-series C', 'Pre-series B',
       'Bridge', 'Series B2', 'Pre- series A', 'Edge', 'Pre-Series B',
       'Seed A', 'Series A-1', 'Seed Funding', 'Pre-seed Round',
       'Seed Round & Series A', 'Pre Series A', 'Pre seed Round',
       'Angel Round', 'Pre series A1', 'Series E2', 'Seed Round',
       'Bridge Round', 'Pre seed round', 'Pre series B', 'Pre series C',
       'Seed Investment', 'Series D1', 'Mid series', 'Series C, D',
       '$1200000', 'Seed+', 'Series F2', 'Series A+', 'Series B3', 'PE',
       'Series F1', 'Pre-series A1', '$300000', 'Early seed', '$6000000',
       '$1000000', 'Seies A', 'Series A2', 'Series I'], dtype=object)
```

We remove the website url from the stage since it is not a stage name

```
data_final['Stage'] = data_final['Stage'].apply(lambda x: str(x).replace('https://docs.google.com/spreadsheets/d/1x9ziNeaz6a
```

We also saw amount in the stage column which we also change. It seen as below

Removing and replacing the mismatch values for instance moneys appears in stage

```
data_final['Stage'] = data_final['Stage'].apply(lambda x: str(x).replace('$6000000', 'Seed Funding'))
data_final['Stage'] = data_final['Stage'].apply(lambda x: str(x).replace('$1000000', 'Seed Funding'))
data_final['Stage'] = data_final['Stage'].apply(lambda x: str(x).replace('$300000', 'Pre seed Round'))
data_final['Stage'] = data_final['Stage'].apply(lambda x: str(x).replace('$1200000', 'Series B3'))
data_final['Stage'] = data_final['Stage'].apply(lambda x: str(x).replace('nan', 'Bridge Round'))
```

We replaced the NaNs in the Investor column with Undisclosed, as seen below

```
data_final['Investor'].unique()
```

```
array([nan, 'Sixth Sense Ventures', 'General Atlantic', ...,
       'Owl Ventures', 'Winter Capital, ETS, Man Capital',
       '3one4 Capital, Kalaari Capital'], dtype=object)
```

```
data_final['Investor'] = data_final['Investor'].apply(lambda x: str(x).replace('nan', 'Undisclosed'))
```

```
missing_percentage=data_final.isna().mean()*100
missing_percentage
```

```
Company Name     0.000000
Sector           0.658376
Stage            0.000000
Amount($)        0.000000
Location         4.169715
Funding Year     0.000000
Founders        14.630578
Investor         0.000000
dtype: float64
```

At the data preparation stage, we inspect the datasets in full, present it, test our hypotheses and rethink the cleaning, and creating new features that will help us answer our question asked in the beginning

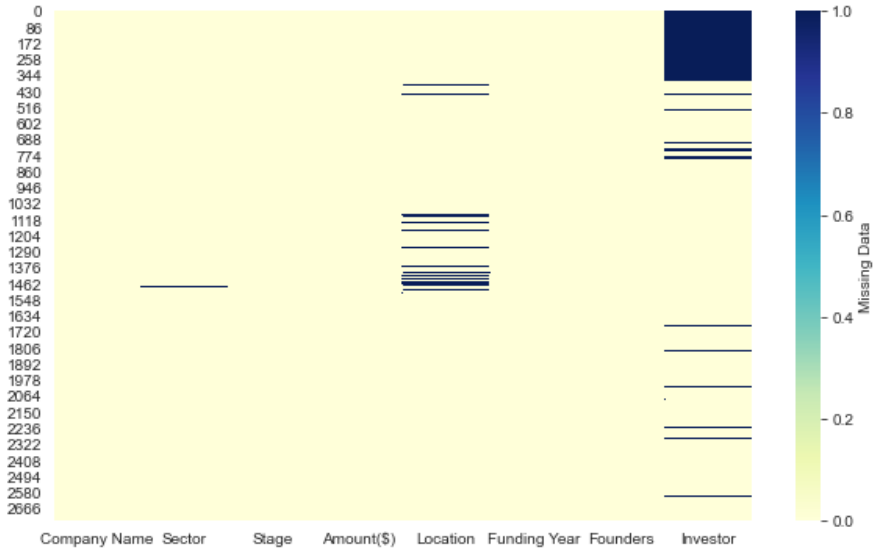
```
1 data_final.head(10)
```

	Company Name	Sector	Stage	Amount($)	Location	Funding Year	Founders	Investor
0	TheCollegeFever	Brand Marketing	Seed	250000.0	Bangalore	2018.0	Arnav Kumar	Undisclosed
1	PayMe India	Financial Services	Angel	2000000.0	Noida	2018.0	Arnav Kumar	Undisclosed
2	Eunimart	E-Commerce Platforms	Seed	1600000.0	Hyderabad	2018.0	Arnav Kumar	Undisclosed
3	Hasura	Cloud Infrastructure	Seed	1600000.0	Bengaluru	2018.0	Arnav Kumar	Undisclosed
4	Freightwalla	Information Services	Seed	1600000.0	Mumbai	2018.0	Arnav Kumar	Undisclosed
5	Microchip Payments	Mobile Payments	Seed	1600000.0	Bangalore	2018.0	Arnav Kumar	Undisclosed
6	BizCrum Infotech Pvt. Ltd.	B2B	Seed	1600000.0	Delhi	2018.0	Arnav Kumar	Undisclosed
7	Emojifi	Internet	Seed	1600000.0	Bengaluru	2018.0	Arnav Kumar	Undisclosed
8	Flock	Apps	Seed	1600000.0	India	2018.0	Arnav Kumar	Undisclosed
9	Freshboxx	Food Delivery	Seed	1600000.0	Hubli	2018.0	Arnav Kumar	Undisclosed

The graph represents where we still missing values. For instance, the investor column and location has most missing values which needs to be worked on.

```
1 plt.figure(figsize=(10,6))
2 sns.heatmap(data_final.isna(),
3             cmap="YlGnBu",
4             cbar_kws={'label': 'Missing Data'})
```

```
61]: <AxesSubplot:>
```



We used the simple imputer method to replace all the missing values with the median in that columns

```
# Find the mode of the 'Investor' column
mode = data_final['Founders'].mode()[0]
```

```
mode
```

```
'Arnav Kumar'
```

```
# Replace missing values in the 'Founders' column with the mode
data_final['Founders'].fillna(mode, inplace=True)
```

```
# Find the mode of the 'Location' column
mode = data_final['Location'].mode()[0]
```

```
mode
```

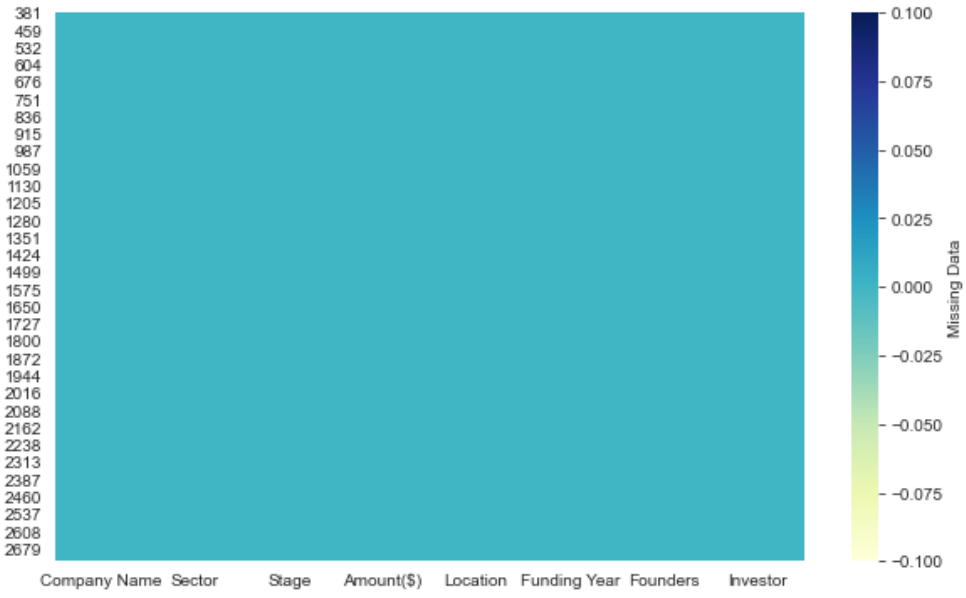
```
'Bangalore'
```

```
# Replace missing values in the 'Location' column with the mode
data_final['Location'].fillna(mode, inplace=True)
```

After repeating the above method for all the columns, we present graph to show if all the missing values has replaced. The image below clearly shows there are no missing values again in our dataset. This way our data is good for our final analysis.

```
plt.figure(figsize=(10,6))
sns.heatmap(data_final.isna(),
            cmap="YlGnBu",
            cbar_kws={'label': 'Missing Data'})
```

```
<AxesSubplot:>
```



Finally, we check our dataset to see how the basic statistics are. The following codes and its output simplifies this process

```
# calculate basic statistical measures
data_final_mean = data_final['Amount($)'].mean()
data_final_median = data_final['Amount($)'].median()
data_final_mode = data_final['Amount($)'].mode()
data_final_std_dev = data_final['Amount($)'].std()
data_final_min_val = data_final['Amount($)'].min()
data_final_max_val = data_final['Amount($)'].max()
```

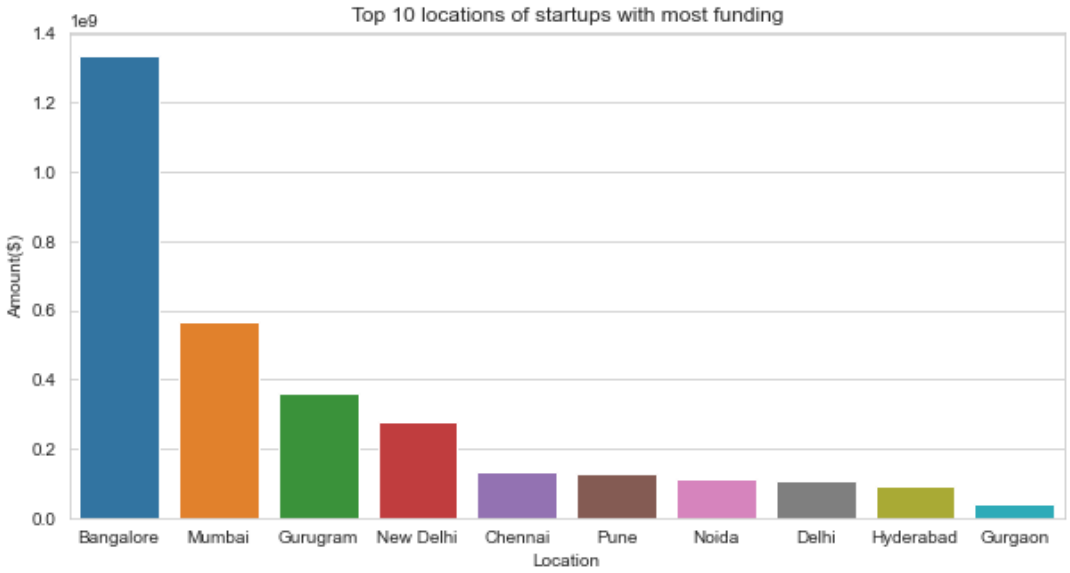
```
print("Mean: ", data_final_mean)
print("Median: ", data_final_median)
print("Mode: ", data_final_mode)
print("Standard Deviation: ", data_final_std_dev)
print("Minimum Value: ", data_final_min_val)
print("Maximum Value: ", data_final_max_val)
```

```
Mean:  1600000.0
Median:  1600000.0
Mode:  0    1600000.0
Name: Amount($), dtype: float64
Standard Deviation:  0.0
Minimum Value:  1600000.0
Maximum Value:  1600000.0
```

QUESTIONS TO BE ANSWERED

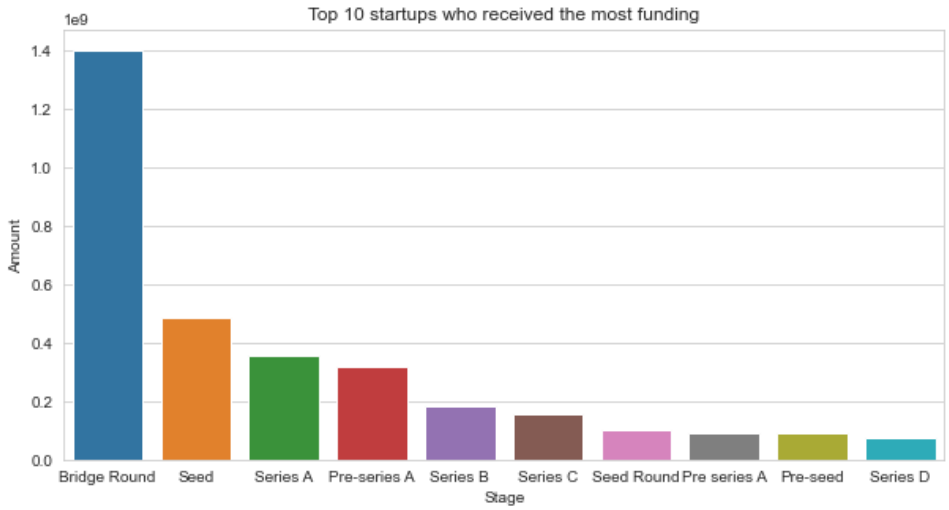
1. Which type of start-up location gets the most funding?

This question seeks to find answers to which startup location receives the most funding in Indian



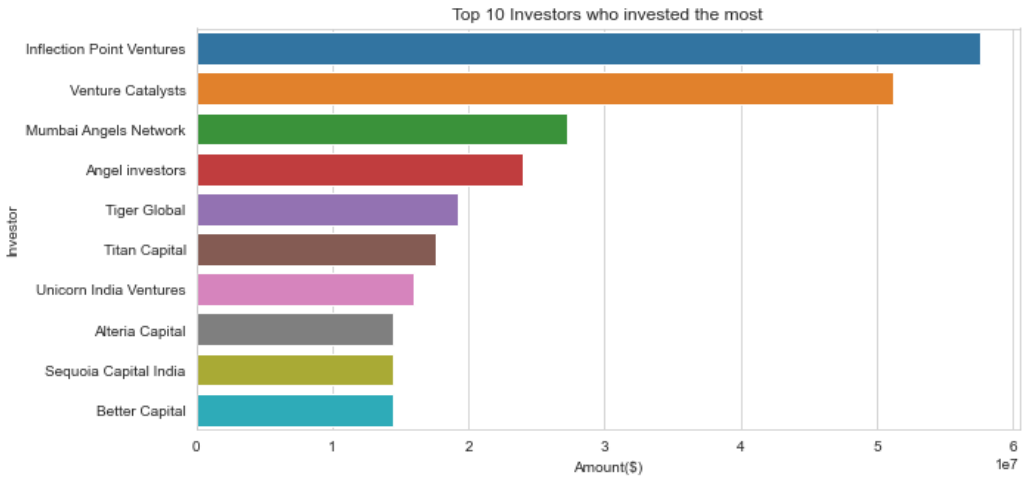
From the visual above we can say that Bangalore received the most funding followed by Mumbai which had less than half of Bangalore's value. It can also be said that the bottom three startups that received the least amount are Delhi, Hyderbad and Gurgaon respectively

2. At which stage do start-ups get more funding from investors?



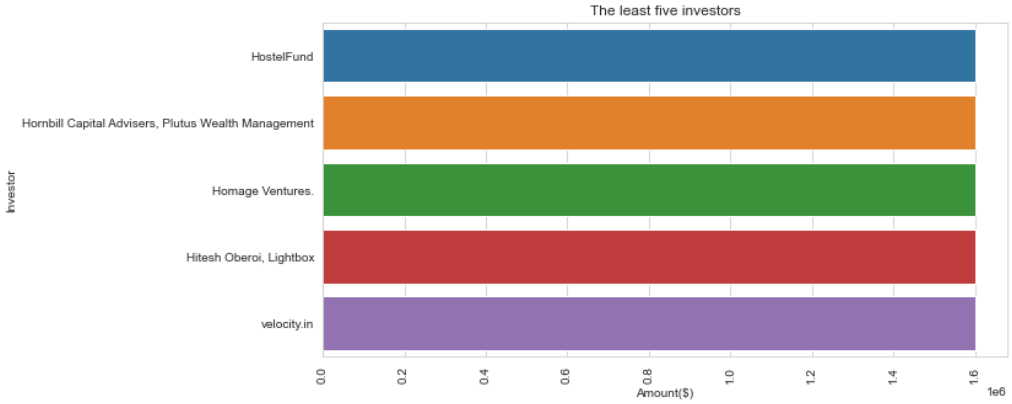
Bridge Round stage received the most funding, Series D had the least funding. It can be concluded that Bridge Round received more than twice the funds received by Seed stage and any other stages.

3. Which type of investors invest the most money?



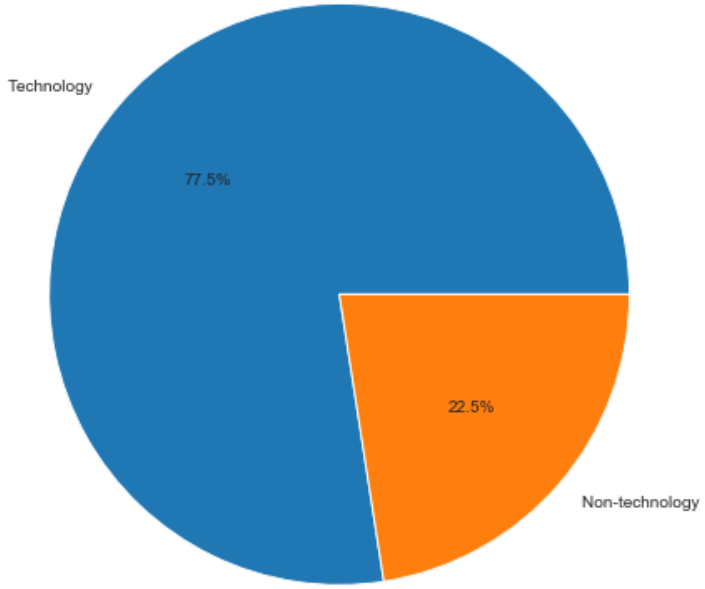
The bar chat above represents the share of how much each investor had invested. Inflection Point Ventures topped the list, followed by Venture. Alteria Capital, Sequoia Capital India and Better Capital invested same amount.

4. **Which type of investors invested the least money?**



5. **What is the percentage of Technology and Non-Technology in the Indian startups?**

To answer this question, we need to define and classify startups that belong to the group of technology and non-technology. We created function and labelled all Technological sectors and non-technological sectors.



The pie chart represents the share these two groups. Technology has about 77% and Non-technology has about 22% as seen in the above pie char. We therefore conclude, Technological sector has more share than non-technological sectors.

RECOMMENDATIONS

India startup ecosystem is very positive and thriving with strong favor toward technological companies but equal opportunities for non-technological sectors.

There are a few large investments in the non-technical industries which are skewing the mean higher while the technical industries have a larger number of smaller investments which are driving up the sum