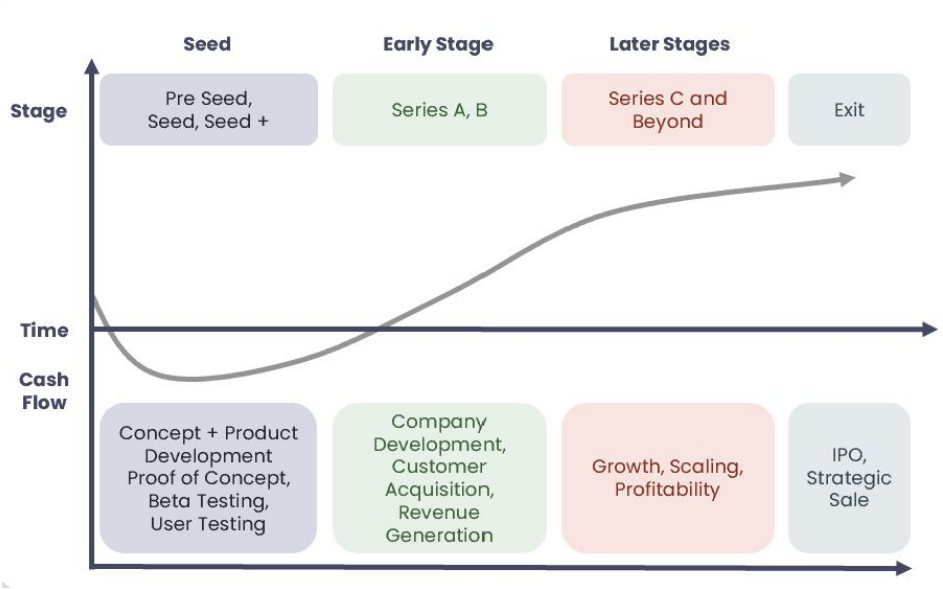
AI-Powered Startup Success Prediction

Group 3 2024/12/04

Intro to Startups & Venture Investment

- A Startup is a Business that :
  - Newly established
  - Solves a problem
  - Grows fast
  - Disrupts the market

- Venture capital (VC) :
  - A type of financing that provides funds to startups.
  - VCs profits by taking high-risk investments in return for high return.



Startup Database on Crunchbase

- Crunchbase:
  - Online platform that provides detailed information about public and private companies, including investment and funding data.
  - Monthly subscription fee needed.

- Breakdowning the Columns :
  - category_list, market
  - country_code, state_code, region, city
  - founded_at, founded_year, status
  - seed, venture, round_A ~ round_H
  - funding_total_usd, funding_rounds, first_funding_at, last_funding_at

01 **Data Cleaning**

02 **Analysis**

03 **Classifying Success**

01

Data Cleaning

0. Data Loading and inspection

- Data Loading and inspection :

```
!pip install pandas seaborn matplotlib plotly numpy catboost category-encoders shap lime
```

- "Status" column :
  - To predict whether a start-up is successful or not, we can rely on classification.
  - Be careful not to overly predict Start-ups as successful than unsuccessful.

```
[ ]  status_counts = df_uncleaned['status'].value_counts(dropna=False)
     check = len(df_uncleaned) - status_counts.get('acquired', 0) - status_counts.get('closed', 0) - status_counts.get('operating', 0)
     missing_values = df_uncleaned["status"].isna().sum()
     other = check - missing_values
```

```
There are 3692 acquired companies
There are 2603 closed companies
There are 41829 operating companies
There are 1314 NaN values
There are 0 other values
```

1. Unnecessary Data & Cleaning columns

- Reduce unnecessary and redundant data :
  - To reduce unnecessary and redundant data, we delete columns that are not relevant for model development. We want to remove post_ipo_equity and post_ipo_debt as well since this they are not interesting to us, as we are only looking at Start-ups.

```
[ ]  df_clean = df_clean.drop(["permalink", "homepage_url", "post_ipo_equity", "post_ipo_debt"], axis=1)
```

- Ensure column names are trimmed and in the correct format :
  - convert 'funding_total_usd' to a float, thereby also transforming "-" to NaN
  - Turning all of our Date Columns into pd.datetime, to ensure consistency and the right data type

```
[ ]  df_clean.columns = df_clean.columns.str.strip()
     df_clean['funding_total_usd'] = pd.to_numeric(df_clean['funding_total_usd'].str.replace(',', ''), errors='coerce')
     df_clean["funding_total_usd"].dtype
```

```
[ ]  df_clean['founded_at'] =  pd.to_datetime(df_clean['founded_at'], format='%Y-%m-%d', errors = 'coerce')
     df_clean['first_funding_at'] =  pd.to_datetime(df_clean['first_funding_at'], format='%Y-%m-%d', errors = 'coerce')
     df_clean['last_funding_at'] =  pd.to_datetime(df_clean['last_funding_at'], format='%Y-%m-%d', errors = 'coerce')
     df_clean['founded_year'] =  pd.to_datetime(df_clean['founded_year'], format='%Y', errors = 'coerce')
     df_clean['founded_month'] =  pd.to_datetime(df_clean['founded_month'], format='%Y-%m', errors = 'coerce')
```

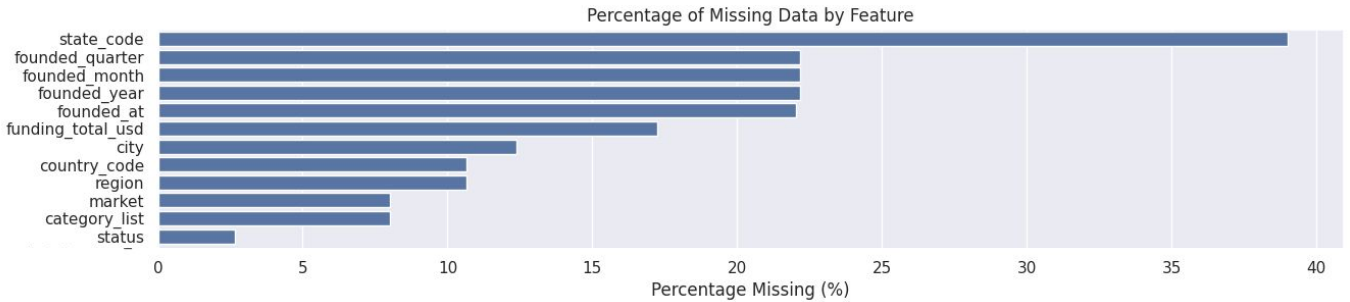
3. Check and drop NaN

- Calculate the number & % of missing values for each column in df_clean
  - Highly Missing Data: State Code, Funding Total USD and City

```
[ ]  missing_counts = df_clean.isnull().sum()

     missing_percentage = (missing_counts / len(df_clean)) * 100

     missing_data = pd.DataFrame({
         'Column': df_clean.columns,
         'Missing Values': missing_counts,
         'Percentage Missing (%)': missing_percentage
     }).sort_values(by='Percentage Missing (%)', ascending=False)
```

=> Geographic Data Gaps: including global companies where such inform is less available.

=> Funding Gaps: quite normal, considering that VC can be highly selective, and only a small percentage of startups secure funding.



4. Data Preparation and Imputation

- Checking STATUS & YEAR column :
  - In our dataset, Status is part of our target variable that can't be imputed, so we drop all NaNs.
  - On the other hand, the founded_year column contains 10,956 missing entries. Imputing founded_year values, is sensible as it preserves the dataset and avoid potential bias.
- Choosing Imputation Over Dropping Values for YEAR column :
  - KNNImputer can estimate missing years by considering the similarity between entries. It assumes that companies founded around similar times may exhibit similar characteristics.
  - To comply with KNNImputer's requirements, it must be converted from datetime to integer format.

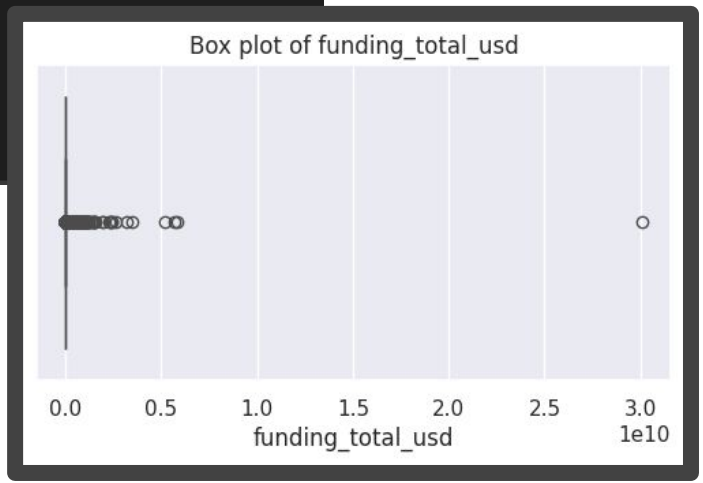
```
[ ]  if not pd.api.types.is_datetime64_any_dtype(df_clean['founded_year']):
         df_clean['founded_year'] = pd.to_datetime(df_clean['founded_year'], errors='coerce', format='%Y')

     print("After conversion, null values:", df_clean['founded_year'].isnull().sum())

     df_clean['founded_year'] = df_clean['founded_year'].dt.year

     imputer = KNNImputer(n_neighbors=5, weights="uniform")
     founded_year_imputed = imputer.fit_transform(df_clean[['founded_year']].astype(float))

     df_clean['founded_year'] = founded_year_imputed.round()

     print("Null values after imputation:", df_clean['founded_year'].isnull().sum())
```

```
After conversion, null values: 10956
Null values after imputation: 0
```

5. Check for outliers

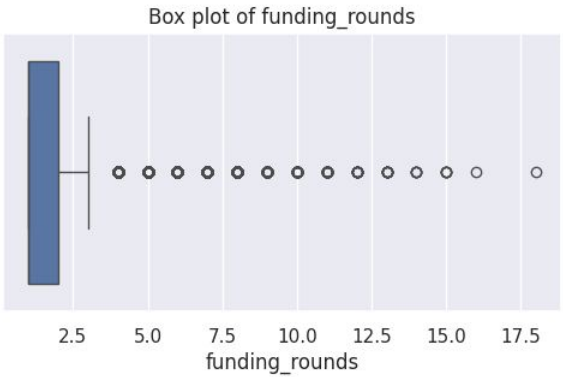
- Focusing on funding rounds :
  - We use box plots to spot any outliers in these important metrics. Since funding data can be pretty varied, some outliers might just be mistakes, while others could be valid extreme values.
  - We apply domain-specific knowledge to figure out and decide whether to remove or tweak these outliers based on how they affect our analysis.

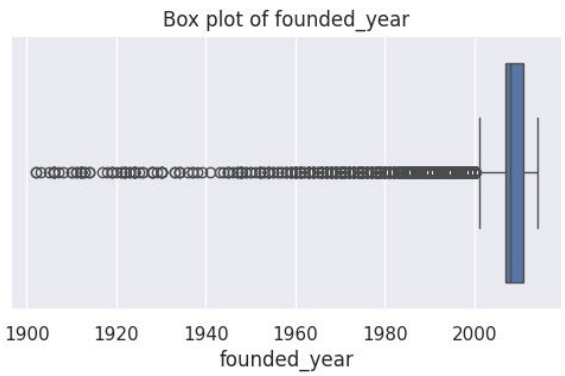
```
numeric_cols = df_clean.select_dtypes(include=['float64', 'int64'])

for col in numeric_cols:
    plt.figure(figsize=(6, 3))
    sns.boxplot(x=df_clean[col])
    plt.title(f'Box plot of {col}')
    plt.show()
```

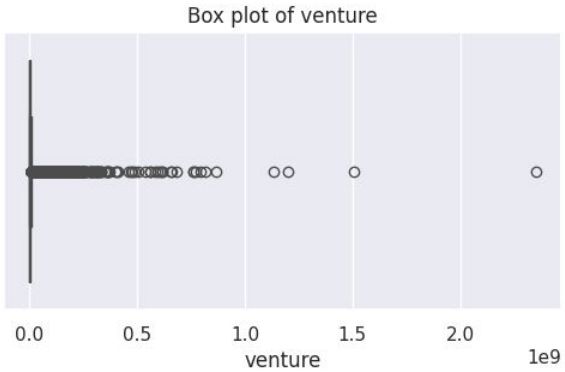


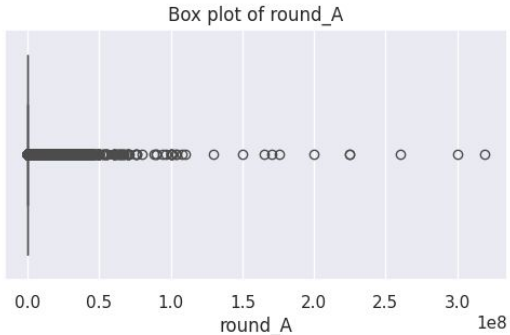
5. Check for outliers

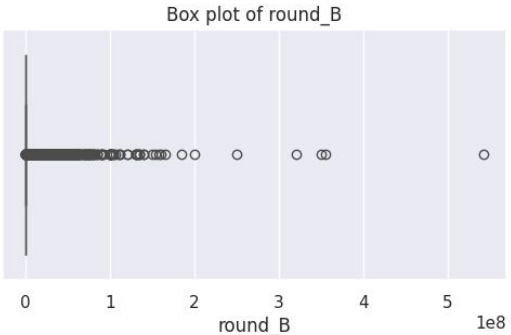
- Most of the data points are concentrated at the lower end, with a long tail stretching out toward the higher values.



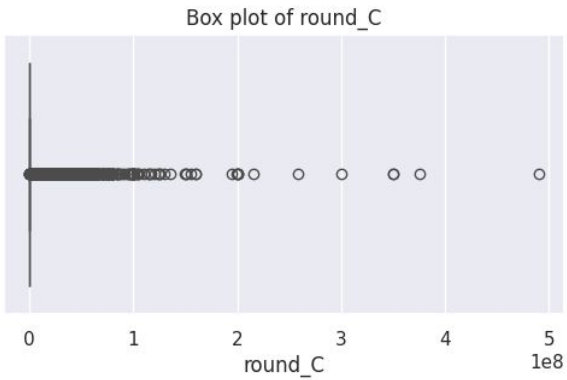


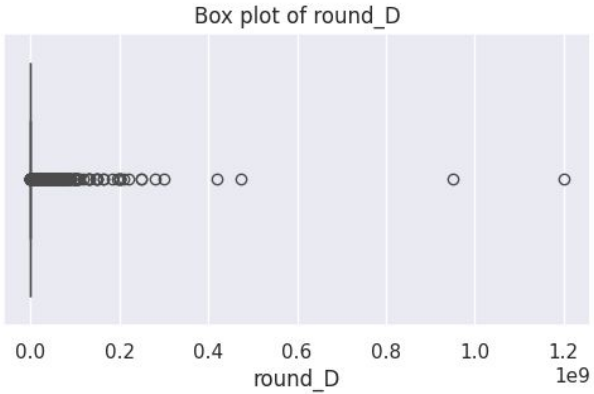


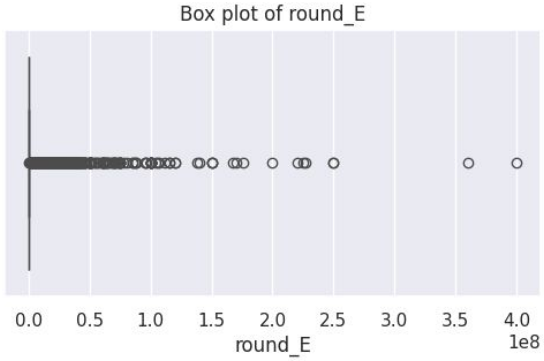


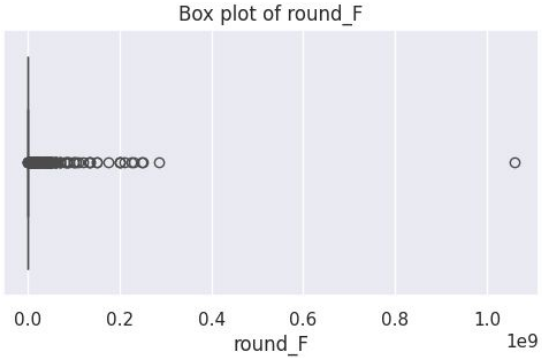


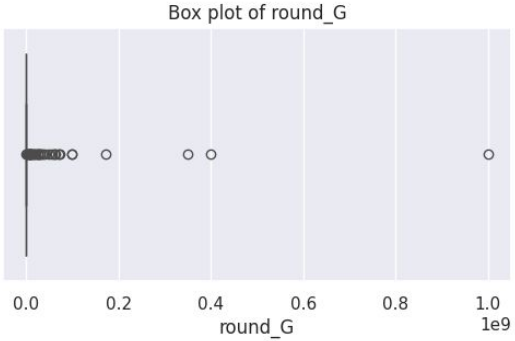
5. Check for outliers

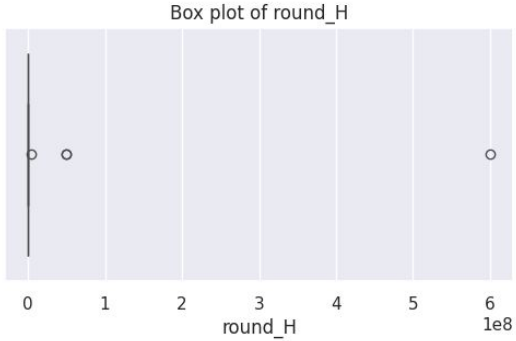












5. Check for outliers

- Box plots results :
  - The funding metrics are skewed. This means there are some outliers and extreme values that can mess with how accurate the predictive model is.
  - We check the dataset to find out why and apply techniques to even things out.

- Set cutoffs for different stages based on industry experience
  - Seed, Angel, and Early Rounds (A, B, C): These stages typically see smaller investment amounts. So we choose cutoffs that reflect common funding ranges, avoiding rare mega-deals distortion.
  - Later Rounds (D, E, F): As startups grow and enter these later stages, the higher cutoffs help us avoid letting a few outsized rounds skew the overall picture.

```python
cutoffs = {
    'venture': 50e6,  # $50 million
    'seed': 10e6,  # $10 million
    'round_A': 20e6,  # $20 million
    'round_B': 30e6,  # $30 million
    'round_C': 40e6,  # $40 million
    'round_D': 60e6,  # $60 million
    'round_E': 80e6,  # $80 million
    'round_F': 100e6,  # $100 million
}
```

```python
original_data_count = df_clean.shape[0]

for funding_type, cutoff in cutoffs.items():
    df_clean = df_clean[df_clean[funding_type] <= cutoff]

print(f"Original Data: {original_data_count} rows")
print(f"Data shape after removing high funding outliers: {df_clean.shape}")
```

```
Original Data: 48124 rows
Data shape after removing high funding outliers: (45702, 35)
```

02

Analysis

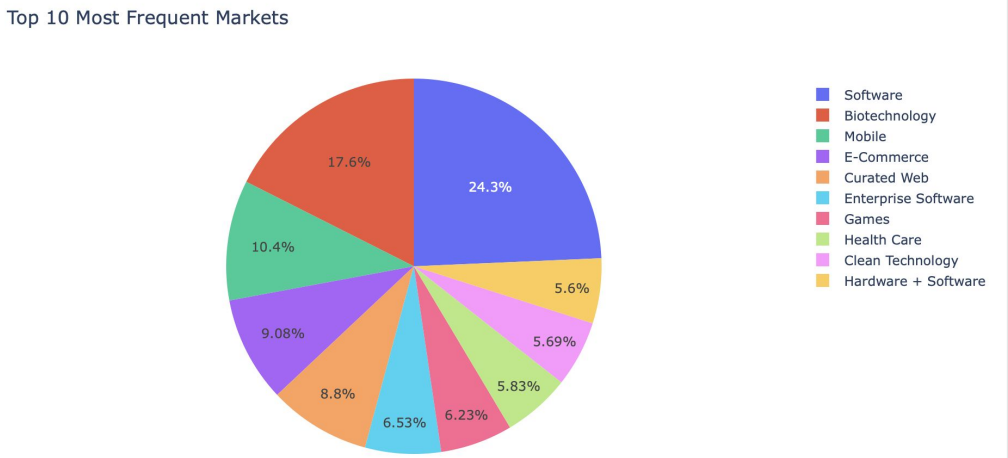
1. Visualization

Bar charts and pie charts are used to display industry distributions, helping to identify which sectors are better and which are more advantageous.

1. Visualization

The code is for generating a pie chart using the Plotly library to visualize the top 10 most frequent markets in the `df_clean['market']` column.

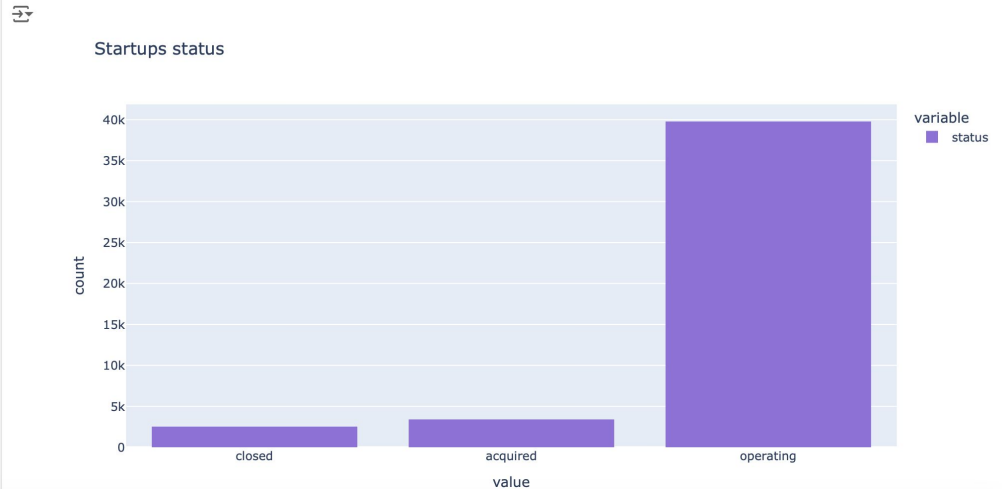
```
top_spheres = df_clean['market'].value_counts()[:10]
fig = px.pie(values=top_spheres, names=top_spheres.index, title= 'Top 10 Most Frequent Markets')
fig.show()
```



1. Visualization

The code is to create a histogram also using Plotly library to visualize the distribution of the ‘Startups status’ in a dataframe called `df_clean`.

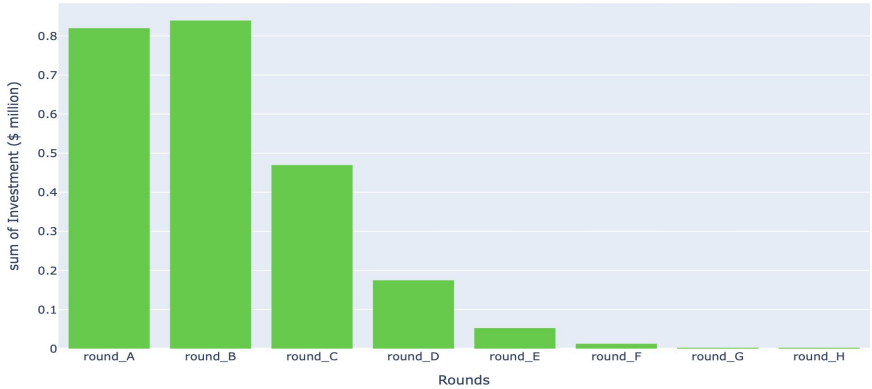
```
fig = px.histogram(df_clean['status'], title='Startups status',
color_discrete_sequence=["mediumpurple"])
fig.update_xaxes(categoryorder='total ascending')
fig.show()
```



1. Visualization

The code snippet is to visualize the average investment across different funding rounds in the `df_clean`DataFrame.

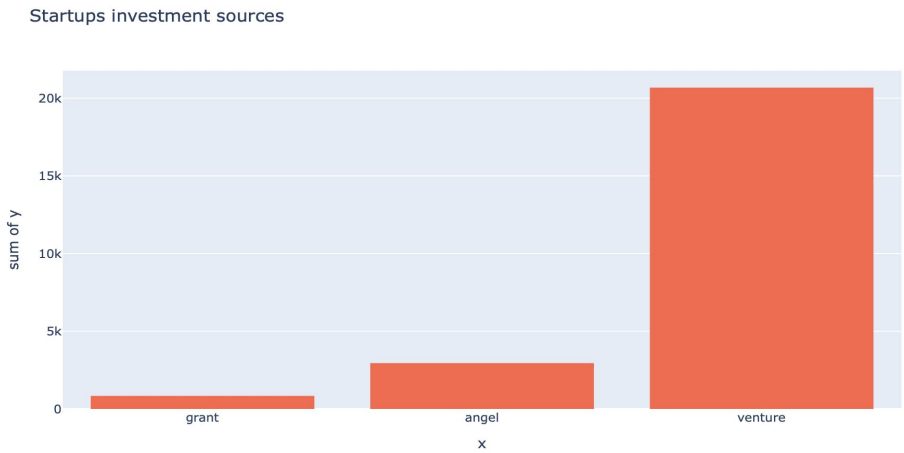
```
rounds = {}
for i in df_clean.columns:
    if 'round_' in i:
        rounds[i] = df_clean[i].mean()/10**6
fig = px.histogram(x=rounds.keys(), y=rounds.values(), labels={'y': 'Investment ($ million)', 'x': 'Rounds'}, color_discrete_sequence=["limegreen"]);
fig.show()
```



1. Visualization

The code is to create a histogram chart to show the count of startups investments from different funding sources: `angel`, `grant`, and `venture`.

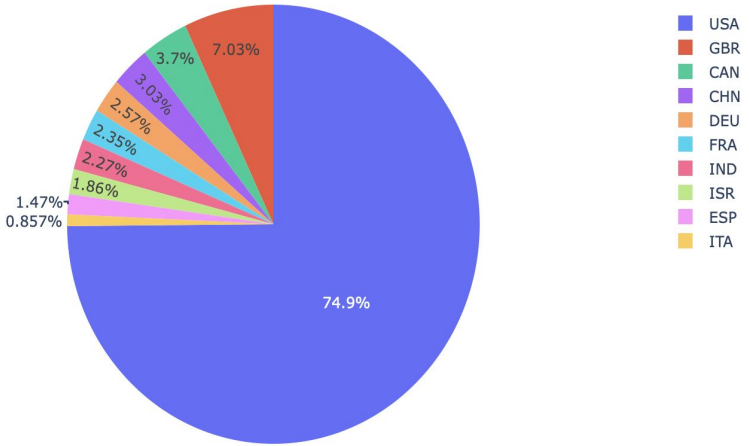
```
fig = px.histogram(x=['angel', 'grant', 'venture'],
y=[len(df_clean[df_clean['angel']!=0]),len(df_clean[df_clean['grant']!=0]),
len(df_clean[df_clean['venture']!=0])],

                   title='Startups investment sources',
                  color_discrete_sequence=["tomato"]);
fig.update_xaxes(categoryorder='total ascending')
fig.show()
```



1. Visualization

The code is to create a pie chart that visualizes the top 10 countries with the most startups based on the country_codecolumn of the df_clean DataFrame.

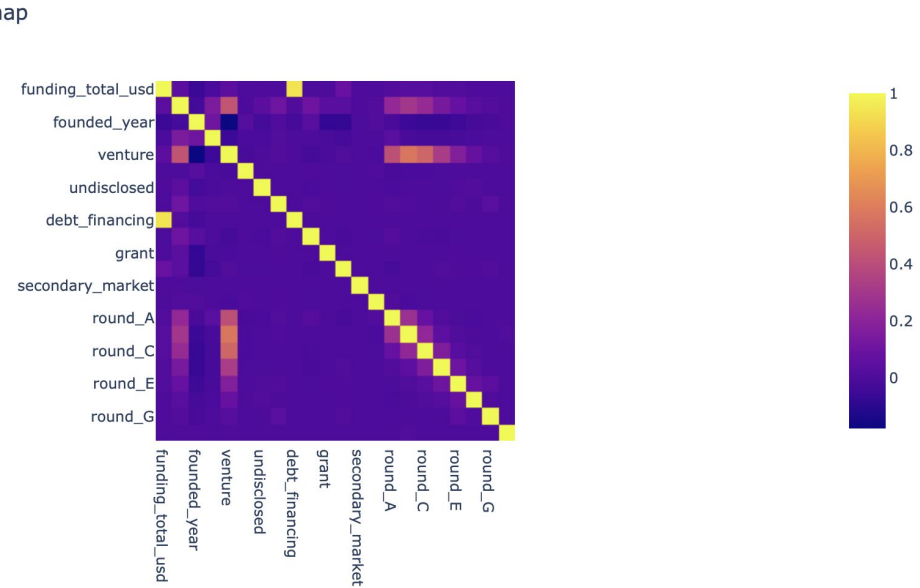
```
top_countries = df_clean['country_code'].value_counts()[:10]
fig = px.pie(values=top_countries, names=top_countries.index, title='Top 10
most popular regions for a startup');
fig.show()
```



2. Correlation

The code generates a heatmap using Plot library to visualize the correlation matrix for numerical columns in `df_clean`.

```
correlation_matrix = df_clean.select_dtypes(include=['number']).corr()
fig = px.imshow(correlation_matrix, title='Correlation heatmap')
fig.show()
```



3. Hypothesis

3.1 Startup will less likely go out of business at less popular market

See dependency between closed and market variables

3.1 Hypothesis

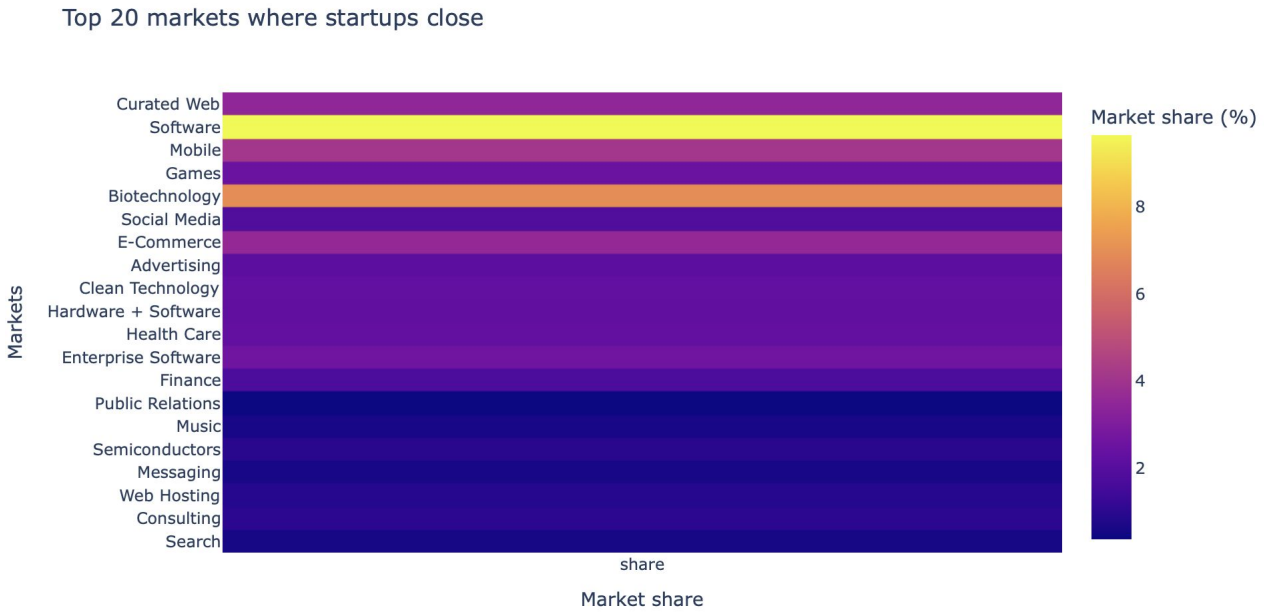
The code is to create a heatmap using Plotly library to visualize the top 20 markets for startups and their market shares.

```
fig = px.imshow(
    markets[['share']][:20],
    aspect='auto',
    title='Top 20 markets where startups close' ,
    labels=dict(x="Market share", y="Markets", color="Market share
(%)"),
)
fig.show()
```

3.1 Hypothesis

Outcome: The closer market to the top, the more 'yellow' its market share becomes.

Choose less popular markets may give us a greater chance to succeed!



3. Hypothesis

3.2 The seed you raise depends on the founded quater

Which quater is the ideal timing to start our first fund raising?

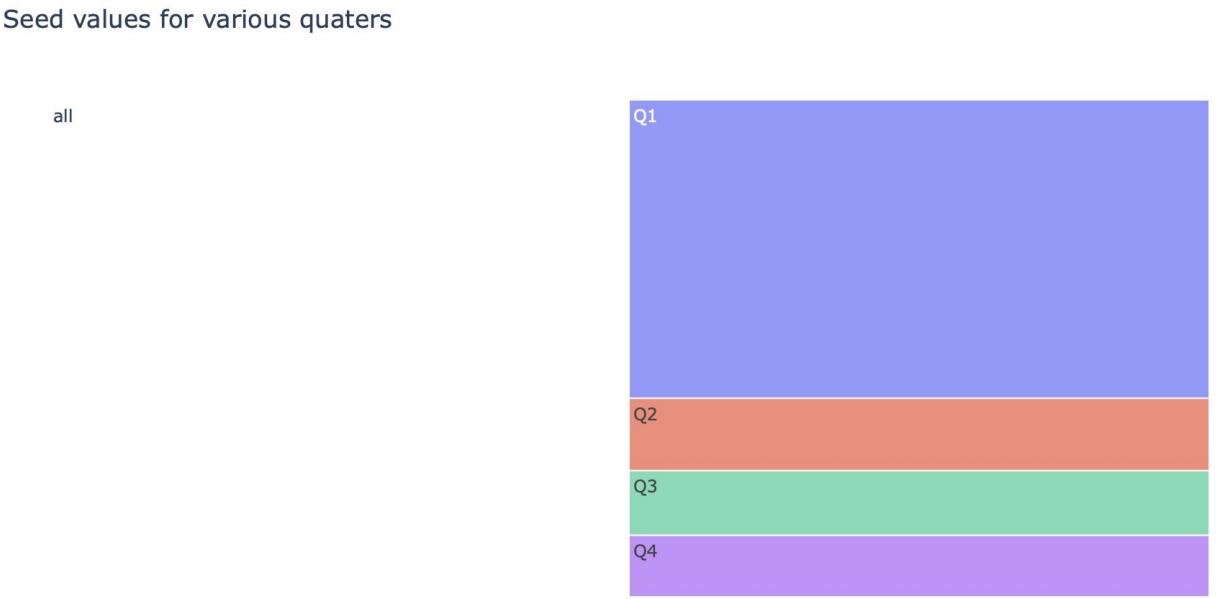
3.2 Hypothesis

The code is to create an chart using Plotly Express to visualize the distribution of seed values across different quarters (`quater`) from the `df_clean` DataFrame.

```
df_clean = df_clean.dropna(subset=['quater'])

df_clean.loc[:, 'seed'] = pd.to_numeric(df_clean['seed'], errors='coerce')

df_clean = df_clean.dropna(subset=['seed'])

fig = px.icicle(
    df_clean,
    path=[px.Constant("all"), 'quater'],
    values='seed',
    title='Seed values for various quaters'
)
fig.show()
```

3.2 Hypothesis

The Q1 is the best timing to startup. Startups should make sure to submit an application for seed funding in the 1st Quater.



4. Data

The code is to explore the descriptive statistics for categorical columns in the DataFrame (`df_clean`).

```
categorical_columns = df_clean.select_dtypes(include=['object']).columns
print("Frequencies of categorical features:\n")
df_clean[categorical_columns].describe().T
```

- **count**: Number of non-null entries.
- **unique**: Number of unique categories.
- **top**: Most frequent category (mode).
- **freq**: Frequency of the most frequent category.

Frequencies of categorical features:

	count	unique	top	freq
name	35538	35494	Roost	3
category_list	33766	14194	|Software|	2738
market	33761	730	Software	3587
status	35539	3	operating	30865
country_code	32662	110	USA	21645
state_code	22661	61	CA	7369
region	32662	990	SF Bay Area	5108
city	32254	3532	San Francisco	2092
founded_quarter	35539	209	2012-Q1	2752
quater	35539	4	Q1	23263

4. Data

The code is also to explore the distribution of numerical features in thedataset.

```
numerical_columns = df_clean.select_dtypes(include=['int64',
'float64']).columns
print("Distribution of numerical features:\n")
df_clean[numerical_columns].describe().T
```

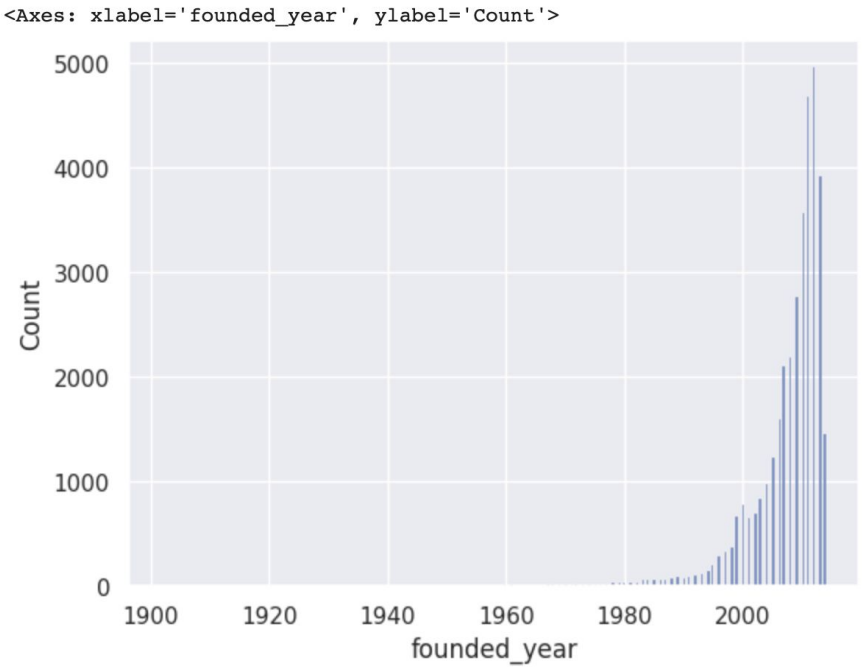
- **count**: Number of non-missing values.
- **mean**: Average value.
- **std**: Standard deviation.
- **min**: Minimum value.
- **25% / 50% / 75%**: Percentiles (1st quartile, median, and 3rd quartile).
- **max**: Maximum value.

	count	mean	std	min	25%	50%	75%	max
funding_total_usd	29350.0	9.745092e+06	1.848672e+08	14.0	300000.0	1600000.0	7250000.0	3.007950e+10
funding_rounds	35539.0	1.666732e+00	1.146566e+00	1.0	1.0	1.0	2.0	1.000000e+01
founded_year	35539.0	2.007613e+03	7.272875e+00	1902.0	2006.0	2010.0	2012.0	2.014000e+03
seed	35539.0	2.436265e+05	6.827684e+05	0.0	0.0	0.0	60000.0	1.000000e+07
venture	35539.0	4.051666e+06	8.482531e+06	0.0	0.0	0.0	3875246.5	5.000000e+07
equity_crowdfunding	35539.0	3.542052e+03	6.242936e+04	0.0	0.0	0.0	0.0	3.000000e+06
undisclosed	35539.0	4.128934e+04	6.102614e+05	0.0	0.0	0.0	0.0	2.000000e+07
convertible_note	35539.0	9.984392e+03	1.409589e+05	0.0	0.0	0.0	0.0	5.000000e+06
debt_financing	35539.0	1.743398e+06	1.612015e+08	0.0	0.0	0.0	0.0	3.007950e+10
angel	35539.0	5.675084e+04	3.031525e+05	0.0	0.0	0.0	0.0	5.000000e+06
grant	35539.0	1.403678e+04	1.878084e+05	0.0	0.0	0.0	0.0	5.000000e+06
private_equity	35539.0	1.067046e+06	1.190450e+07	0.0	0.0	0.0	0.0	3.000000e+08
secondary_market	35539.0	2.810597e+03	3.432911e+05	0.0	0.0	0.0	0.0	6.000000e+07
product_crowdfunding	35539.0	2.523130e+03	5.945797e+04	0.0	0.0	0.0	0.0	3.000000e+06
round_A	35539.0	8.673897e+05	2.506124e+06	0.0	0.0	0.0	0.0	2.000000e+07
round_B	35539.0	9.212252e+05	3.405509e+06	0.0	0.0	0.0	0.0	3.000000e+07
round_C	35539.0	5.171665e+05	2.927384e+06	0.0	0.0	0.0	0.0	4.000000e+07
round_D	35539.0	1.928061e+05	1.919932e+06	0.0	0.0	0.0	0.0	5.000000e+07

5. Time related Features

The code is designed to create a histogram of the `founded_year` column from `df_clean` using Seaborn's `histplot` function.

```
import seaborn as sns

sns.histplot(df_clean['founded_year'].dropna(), kde=False)
```



5. Time related Features

The code filters out rows where the `founded_year` is earlier than 1990, and it provides a summary of how the dataset size changes before and after the filtering.

```
print("Data shape before removing years before 1990:", df_clean.shape)
df_clean = df_clean[df_clean['founded_year'] >= 1990]
print("Data shape after removing years before 1990:", df_clean.shape)
```

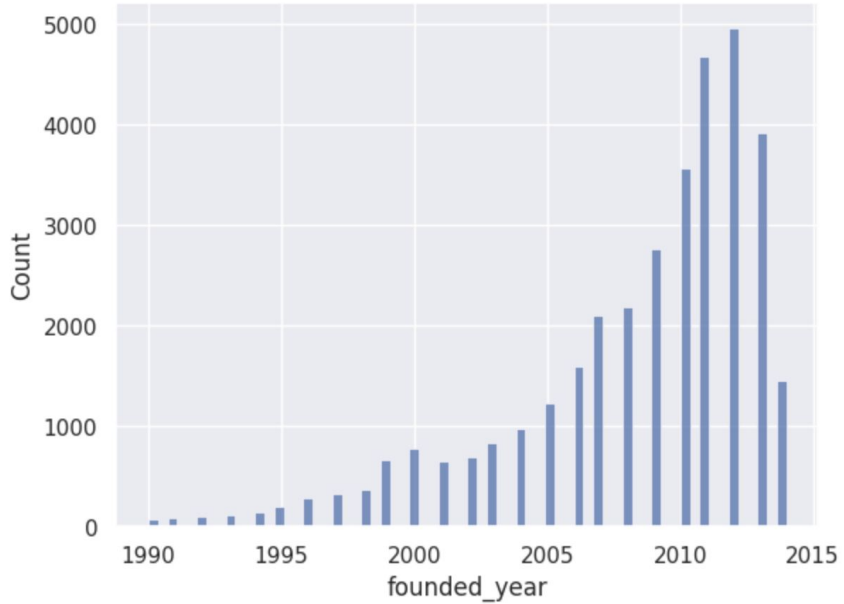
```
Data shape before removing years before 1990: (35539, 36)
Data shape after removing years before 1990: (34690, 36)
```

5. Time related Features

The same program as former one. This is the graph after removing some datas

```
sns.histplot(df_clean['founded_year'].dropna(), kde=False)
```

```
<Axes: xlabel='founded_year', ylabel='Count'>
```



5. Time related Features

We use IQR and Z-score methods to remove outliers in the founding year, ensuring the dataset reflects current, relevant startups and enhances analysis accuracy.

5. Time related Features

Convert datetime to ordinal

Compute Q1, Q3, and IQR

Define bounds for outliers

Count outliers using IQR

Count outliers using Z-scores

Applying the function to each column

```
def calculate_datetime_outliers(df, column_name):
    df[column_name + '_ordinal'] = df[column_name].map(lambda x: x.toordinal() if pd.notnull(x) else np.nan)
    Q1 = df[column_name + '_ordinal'].quantile(0.25)
    Q3 = df[column_name + '_ordinal'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    lower_outliers_IQR_count = df[df[column_name + '_ordinal'] < lower_bound].shape[0]
    upper_outliers_IQR_count = df[df[column_name + '_ordinal'] > upper_bound].shape[0]
    z_scores = np.abs(stats.zscore(df[column_name + '_ordinal'].dropna()))
    outliers_Z_count = np.sum(z_scores > 3)
    print(f"Lower outliers count for {column_name} (IQR method): {lower_outliers_IQR_count}")
    print(f"Upper outliers count for {column_name} (IQR method): {upper_outliers_IQR_count}")
    print(f"Outliers count for {column_name} (Z-score method): {outliers_Z_count}")
datetime_columns = ['founded_at', 'founded_month', 'first_funding_at', 'last_funding_at']
for column in datetime_columns:
    if column in df_clean.columns:
        calculate_datetime_outliers(df_clean, column)
```

5. Time related Features

Output:

```
Lower outliers count for founded_at (IQR method): 1621
Upper outliers count for founded_at (IQR method): 0
Outliers count for founded_at (Z-score method): 501
Lower outliers count for founded_month (IQR method): 1621
Upper outliers count for founded_month (IQR method): 0
Outliers count for founded_month (Z-score method): 501
Lower outliers count for first_funding_at (IQR method): 586
Upper outliers count for first_funding_at (IQR method): 2
Outliers count for first_funding_at (Z-score method): 397
Lower outliers count for last_funding_at (IQR method): 1184
Upper outliers count for last_funding_at (IQR method): 0
Outliers count for last_funding_at (Z-score method): 341
```

5. Time related Features

After already having dropped the outliers for funding_year before, we use z-scores for dropping outliers in datetime data to provide a statistically rigorous approach that considers the distribution of the data and effectively handles extreme values.

5. Time related Features

Drop outliers for datetime columns using z-score
Convert datetime to ordinal
Calculate z-scores
Find outliers based on z-score threshold
Drop rows with outliers
Calculate and display number of rows dropped
Display shape before dropping outliers (overall)
Applying the function to each datetime column
Display shape after dropping outliers (overall)

```
def drop_datetime_outliers(df, column_name, z_threshold=3):
    df[column_name + '_ordinal'] = df[column_name].map(lambda x:
x.toordinal() if pd.notnull(x) else np.nan)
    z_scores = np.abs(stats.zscore(df[column_name + '_ordinal'].dropna()))
    outliers_indices = np.where(z_scores > z_threshold)[0]
    df.drop(df.index[outliers_indices], inplace=True)
    num_rows_dropped = len(outliers_indices)
    print(f"Number of rows dropped for {column_name}: {num_rows_dropped}")
print(f"Shape before dropping outliers (overall):{df_clean.shape}")
datetime_columns = ['founded_at', 'founded_month', 'first_funding_at',
'last_funding_at']
for column in datetime_columns:
    if column in df_clean.columns:
        drop_datetime_outliers(df_clean, column)
print(f"Shape after dropping outliers (overall):{df_clean.shape}")
```

5. Time related Features

Output:

```
Shape before dropping outliers (overall): (34690, 40)
Number of rows dropped for founded_at: 501
Number of rows dropped for founded_month: 197
Number of rows dropped for first_funding_at: 355
Number of rows dropped for last_funding_at: 259
Shape after dropping outliers (overall): (33378, 40)
```

6. Mapping

- **Defining Industry Categories**: Startups were mapped to predefined industry categories such as "Technology," "Healthcare," "Finance," and "E-commerce."
- **Automated Mapping Using Keywords**: Keywords in the company description (`description`) were matched with industry-specific terms to classify companies automatically. For example, keywords like "AI" and "blockchain" were mapped to "Technology."
- **Multi-Industry Mapping**: Companies operating across multiple sectors were assigned to several industry categories to provide a holistic view of their involvement.

6. Mapping

We load the related data first. Exp:

```
admin_services = str('Employer Benefits Programs, Human Resource Automation, Corporate IT, D
advertising = str('Creative Industries, Promotional, Advertising Ad Exchange, Ad Network, Ad
agriculture = str('Agriculture, AgTech, Animal Feed, Aquaculture, Equestrian, Farming, Fores
app = str('Application Performance Monitoring, App Stores, Application Platforms, Enterprise
artificial_intelli = str('Artificial Intelligence, Intelligent Systems, Machine Learning, Na
biotechnology = str('Synthetic Biology, Bio-Pharm, Bioinformatics, Biometrics, Biopharma, Bi
clothing = str('Fashion, Laundry and Dry-cleaning, Lingerie, Shoes').split(', ')
shopping = str('Consumer Behavior, Customer Support Tools, Discounts, Reviews and Recommenda
community = str("Self Development, Sex, Forums, Match-Making, Babies, Identity, Women, Kids,
electronics  = str('Mac, iPod Touch, Tablets, iPad, iPhone, Computer, Consumer Electronics,
consumer_goods= str('Commodities, Sunglasses, Groceries, Batteries, Cars, Beauty, Comics, Co
content = str('E-Books, MicroBlogging, Opinions, Blogging Platforms, Content Delivery Networ
data = str('Optimization, A/B Testing, Analytics, Application Performance Management, Artifi
design = str('Visualization, Graphics, Design, Designers, CAD, Consumer Research, Data Visua
education = str('Universities, College Campuses, University Students, High Schools, All Stud
energy = str('Gas, Natural Gas Uses, Oil, Oil & Gas, Battery, Biofuel, Biomass Energy, Clean
events = str('Concerts, Event Management, Event Promotion, Events, Nightclubs, Nightlife, Re
financial = str('Debt Collecting, P2P Money Transfer, Investment Management, Trading, Accoun
food = str('Specialty Foods, Bakery, Brewing, Cannabis, Catering, Coffee, Confectionery, Coo
gaming = str('Game, Games, Casual Games, Console Games, Contests, Fantasy Sports, Gambling,
government = str('Polling, Governance, CivicTech, Government, GovTech, Law Enforcement, Mili
hardware= str('Cable, 3D, 3D Technology, Application Specific Integrated Circuit (ASIC), Aug
health_care = str('Senior Health, Physicians, Electronic Health Records, Doctors, Healthcare
it = str('Distributors, Algorithms, ICT, M2M, Technology, Business Information Systems, Civi
internet = str('Online Identity, Cyber, Portals, Web Presence Management, Domains, Tracking,
invest = str('Angel Investment, Banking, Commercial Lending, Consumer Lending, Credit, Credi
manufacturing = str('Innovation Engineering, Civil Engineers, Heavy Industry, Engineering Fi
```

6. Mapping

The code defines a function to categorize a given market string into an industry group based on predefined keywords.

```
def get_industry_group(market_string):
    if not isinstance(market_string, str):
        return "Unknown"
    for industry, keywords in industry_keywords.items():

        pattern = '|'.join(map(re.escape, keywords))
        if re.search(pattern, market_string, flags=re.IGNORECASE):
            return industry
    return "Other"

df_features['Industry_Group'] = df_features['market'].apply(get_industry_group)
```

6. Mapping

```
print("Number of unique industries:", df_features['Industry_Group'].nunique())
```

Outcome: Number of unique industries: 46

```
df_features.groupby('Industry_Group').size().sort_values(ascending = False).reset_index()
```

	Industry_Group	0
0	Software	6447
1	Internet	2087
2	Health Care	2010
3	Shopping	1956
4	Biotechnology	1933
5	Unknown	1590
6	Community	1527
7	Mobile	1508
8	Advertising	1188
9	Financial	1001
10	Gaming	983
11	Services	816
12	Data	789
13	Education	749
14	Other	730
15	Content	704

6. Mapping

The code merges two DataFrames, `df_features` and `country`, based on a common column (`country_code`in `df_features` and `alpha-3` in `country`).

```
country = pd.read_csv('continents.csv')
country.rename(columns = {"region": "continent"}, inplace = True)
country = country[['continent', 'alpha-3']]
df_features_merged = df_features.merge(country, left_on='country_code',
right_on='alpha-3', how= "left", suffixes=('_left', '_right'))
```

6. Mapping

The code groups the `df_features_merged` DataFrame by the `continent` column and calculates the size (i.e., the number of rows) in each group.

```
df_features_merged.groupby('continent').size()
```

continent	0
Africa	138
Americas	21782
Asia	2789
Europe	5608
Oceania	273

dtype: int64

03

Classifying Success

1.Binning

- Objective :
  - Categorize total funding amounts into groups such as "low," "medium-low," "medium-high," and "high.“
- Advantages :
  - Simplifies analysis by reducing variability across a wide range of funding amounts.
  - Provides a clear comparison of startups' funding stages without diving into specific dollar values.

1.Binning

- Define a function to calculate the total funding amount :

```
def calculate_funding_total(df):
    df["funding_total_usd"].fillna(0, inplace=True)

    columns_to_sum = ['funding_rounds', 'seed', 'venture', 'equity_crowdfunding', 'undisclosed',
                      'convertible_note', 'angel', 'grant', 'private_equity', 'secondary_market',
                      'product_crowdfunding', 'round_A', 'round_B', 'round_C', 'round_D',
                      'round_E', 'round_F', 'round_G', 'round_H']

    df["funding_total_usd"] = df[columns_to_sum].sum(axis=1)

calculate_funding_total(df_features_merged)
```

- Funding amount categorization (Binning) :
    - Use pd.qcut divides data into bins with equal proportions of observations, based on the actual data distribution.

```
group_labels_1 = ["low", "medium-low", "medium-high", "high"]

df_features_merged["category_total"] = pd.qcut(df_features_merged["funding_total_usd"], 4, labels = group_labels_1)
```

1.Binning

- Calculate the average time between funding rounds:

```
df_features_merged['funding_duration_days'] = (df_features_merged['last_funding_at'] - df_features_merged['first_funding_at']).dt.days

df_features_merged['avg_days_between_round'] = df_features_merged['funding_duration_days'] / df_features_merged['funding_rounds']

df_features_merged["avg_years_between_round"] = df_features_merged['avg_days_between_round'] / 365
```

- Average amount raised per funding round:

```
df_features_merged["avg_raised_amount_usd"] = df_features_merged["funding_total_usd"]/df_features_merged["funding_rounds"]
```

2.Target Variable

- Classifying Successful Startups:
  1. Acquired: Directly classified as successful.
  2. Operating for more than 5 years
     && without G、H funding occurred.
  3. Operating for less than 5 years
     && funding category is not “low” .
     - (funding category is only “high”)

	status	category_total	years_operating	Target
0	acquired	medium-high	2.499658	1
1	operating	low	2.097194	0
2	operating	medium-high	3.668720	1
3	operating	medium-low	0.914442	1
5	operating	low	4.914442	0
...	...	...	...	...
33373	operating	high	3.003422	1
33374	operating	medium-high	7.915127	1
33375	operating	medium-low	1.839836	1
33376	operating	medium-low	2.551677	1
33377	operating	high	15.915127	1

```python
def classify_status(row):
    if row['status'] == 'acquired':
        return 1
    elif row['status'] == 'operating':
        if row['years_operating'] > 5 and row['round_G'] == 0 and row['round_H'] == 0:
            return 1
        elif row['years_operating'] <= 5:
            if row['category_total'] == 'high' or row['category_total'] == 'medium-high' or row['category_total'] == 'medium-low':
                return 1
            else:
                return 0
    return None
```

3.Training & Testing

- Load Data
  - Use pandas.read_csv() to load the data and convert it into a DataFrame format.

```
# 1. 載入數據 / Load data
def load_data(file_path):
    print("Loading data...")
    data = pd.read_csv(file_path)
    print("Data loaded successfully!")
    return data
```

3.Training & Testing

- Process categorical variables :
  - Method：Use pandas.get_dummies( ) making One-Hot Encoding。

```
# 2. 處理類別變數 / Process categorical variables
def preprocess_data(data):
    print("Converting categorical columns to One-Hot Encoding...")
    categorical_columns = data.select_dtypes(include=['object']).columns
    data = pd.get_dummies(data, columns=categorical_columns, drop_first=False)
    print("Categorical conversion completed!")
    return data
```

3.Training & Testing

- Process categorical variables :
  - One-Hot Encoding converts categorical data into binary columns for machine learning models to process non-numeric data.

Industry_Group	Continent
Content	Americas
Content	Europe
Electronics	Europe

Industry_group_Content	Industry_group_Electronics	Continent_Americas	Continent_Europe
1	0	1	0
1	0	0	1
0	1	0	1

3.Training & Testing

- Data Splitting：
  - Use sklearn.model_selection.train_test_split。
  - Split the data into feature sets (Features) and target variables (Target), and further divide them into training and testing sets.

```python
# 3. 切分數據為訓練集和測試集 / Split data into training and testing sets
def split_data(data, features, target_col="Target"):
    print("Splitting data into training and testing sets...")
    x = data[features]
    y = data[target_col]
    x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1, test_size=0.2)
    return x_train, x_test, y_train, y_test
```

3.Training & Testing

- Model Training :
  - Random Forest model

```
# 4. 訓練隨機森林模型 / Train a Random Forest model
def train_rf(x_train, y_train, n_estimators=100):
    clf = RandomForestClassifier(n_estimators=n_estimators, random_state=42)
    clf.fit(x_train, y_train)
    return clf
```

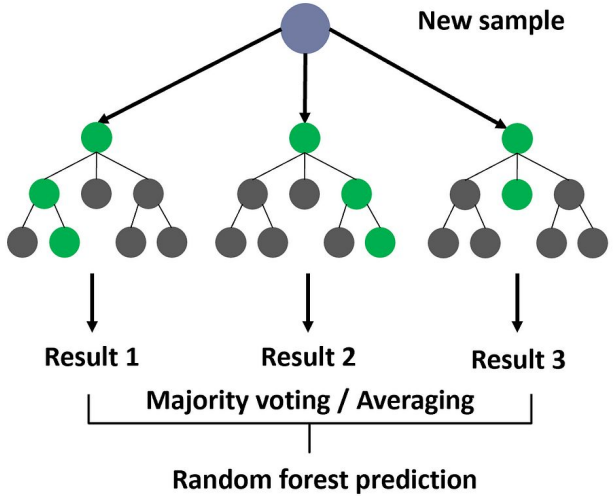
  - K-Nearest Neighbors (KNN) model

```
# 5. 訓練KNN模型 / Train a K-Nearest Neighbors (KNN) model
def train_knn(x_train, y_train, n_neighbors=3):
    model = KNeighborsClassifier(n_neighbors=n_neighbors)
    model.fit(x_train, y_train)
    return model
```

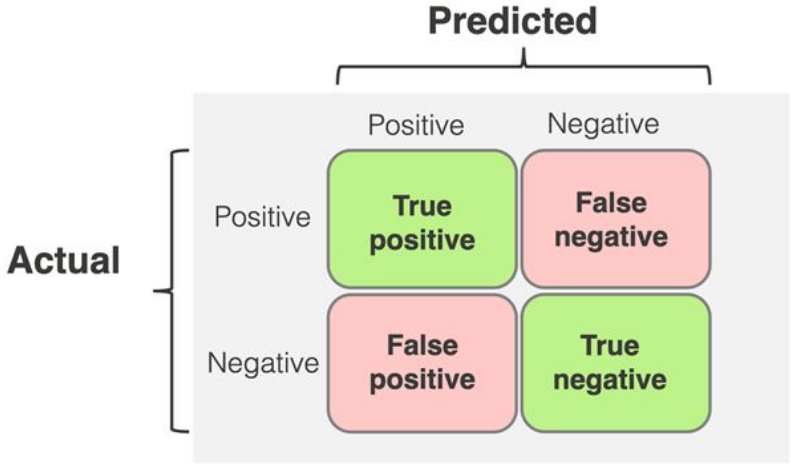
3.Training & Testing

- Random Forest model

1. High Stability :
   - majority vote
2. Handles Diversity
   - Random data
3. Versatile Usage
   - classification or regression tasks.



3.Training & Testing



- Confusion Matrix :
  - Use sklearn.metrics.confusion_matrix

```
# 7. 混淆矩陣 / confusion matrix
def plot_confusion_matrix(y_test, y_pred):
    cm = confusion_matrix(y_test, y_pred)
    tn, fp, fn, tp = cm.ravel()  # 展開混淆矩陣中的值 / Unpack confusion matrix values
    print(f"True Positives (TP): {tp} - 預測成功且實際成功 / Predicted success and actually success")
    print(f"True Negatives (TN): {tn} - 預測失敗且實際失敗 / Predicted failure and actually failure")
    print(f"False Positives (FP): {fp} - 預測成功但實際失敗 / Predicted success but actually failure")
    print(f"False Negatives (FN): {fn} - 預測失敗但實際成功 / Predicted failure but actually success")
```

3.Training & Testing

- Evaluate Model Performance :
  - Use Metric Functions from sklearn.metrics Module

```python
# 6. 評估模型性能 / Evaluate model performance
def evaluate_model(model, x_test, y_test):
    predictions = model.predict(x_test)
    accuracy = accuracy_score(y_test, predictions)
    precision = precision_score(y_test, predictions, average='weighted')
    recall = recall_score(y_test, predictions, average='weighted')
    f1 = f1_score(y_test, predictions, average='weighted')
```

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.Training & Testing

- Displaying Results :

```
-------------------------------------
Evaluating Random Forest model performance...
Accuracy: 0.9953
Precision: 0.9954
Recall: 0.9953
F1 Score: 0.9954
True Positives (TP): 4831 - 預測成功且實際成功 / Predicted success and actually success
True Negatives (TN): 936 - 預測失敗且實際失敗 / Predicted failure and actually failure
False Positives (FP): 4 - 預測成功但實際失敗 / Predicted success but actually failure
False Negatives (FN): 23 - 預測失敗但實際成功 / Predicted failure but actually success
-------------------------------------
Evaluating KNN model performance...
Accuracy: 0.9815
Precision: 0.9821
Recall: 0.9815
F1 Score: 0.9817
True Positives (TP): 4776 - 預測成功且實際成功 / Predicted success and actually success
True Negatives (TN): 911 - 預測失敗且實際失敗 / Predicted failure and actually failure
False Positives (FP): 29 - 預測成功但實際失敗 / Predicted success but actually failure
False Negatives (FN): 78 - 預測失敗但實際成功 / Predicted failure but actually success
```

3.Training & Testing

3. Operating for less than 5 years
   && funding category is not “low” .
   - (funding category is only “high”)

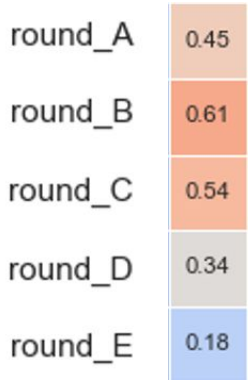
- Model Performance Comparison With Different Categories_total
  - Category_total ≠ "low“ for success classify is much better

Model	category_total ≠ "low"	category_total = "high"
Random Forest	0.9953 >	0.9852
KNN	0.9815 >	0.8687

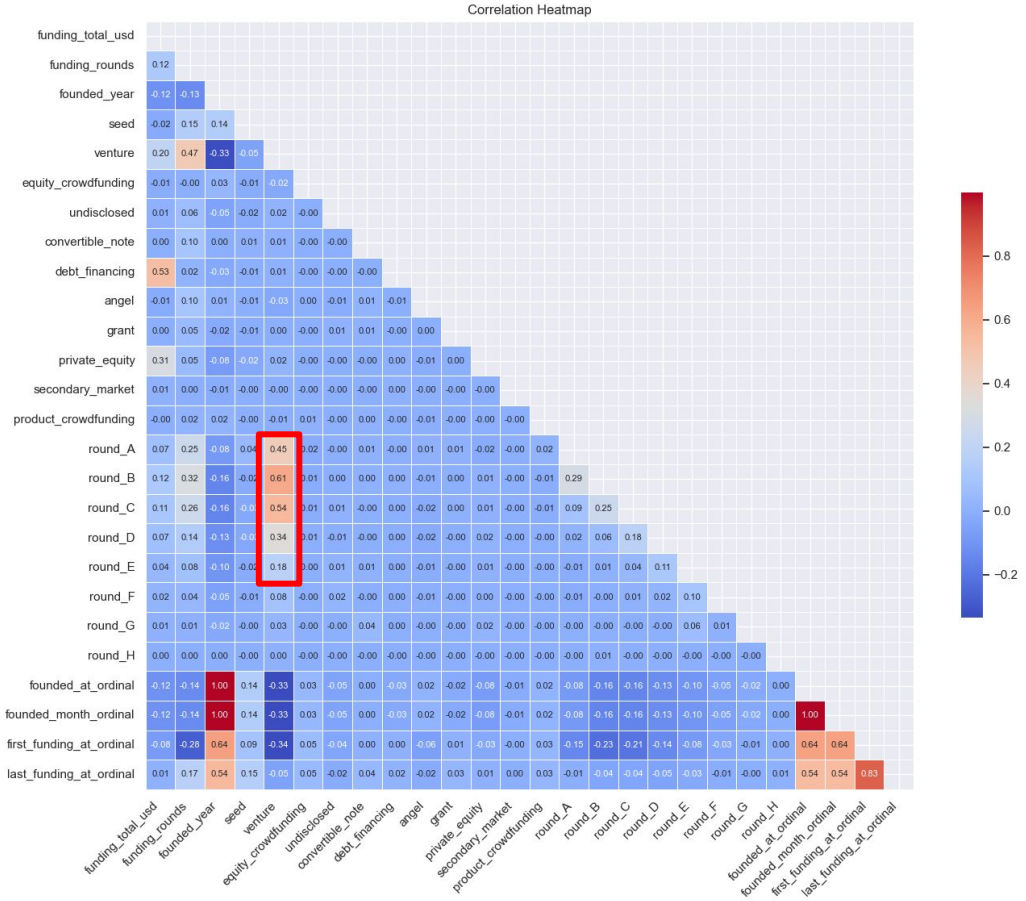
  - Random Forest model perform better than Knn model in both side.

4.Correlation

- Correlation Heatmap Analysis

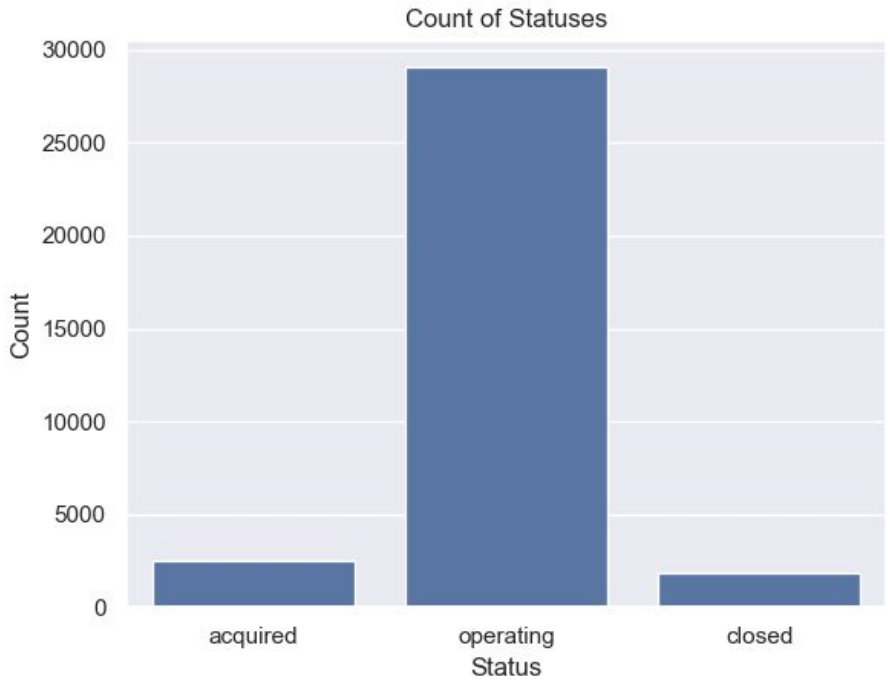


venture



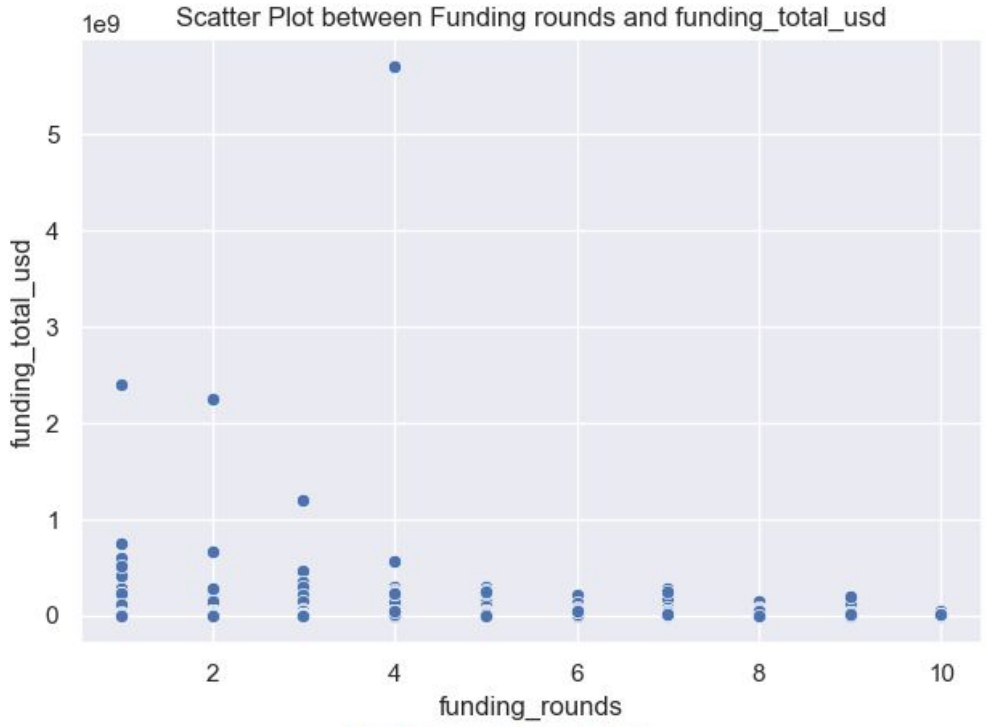
4.Correlation

- Breakdown of Company Status



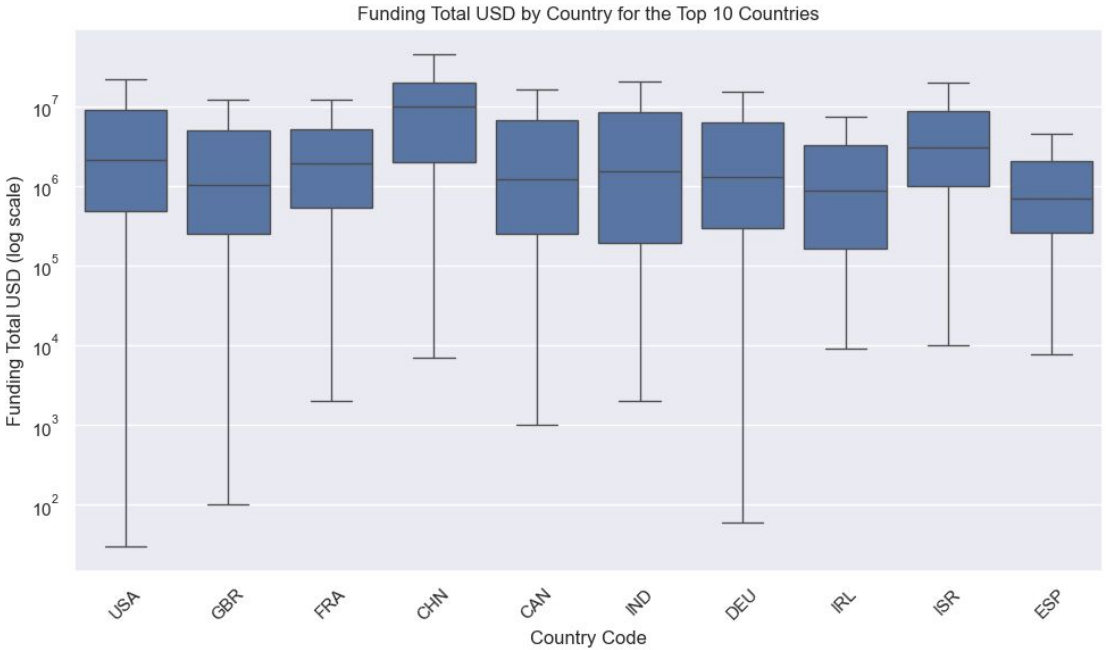
4.Correlation

- Relationship Between Funding Rounds and Total Funding



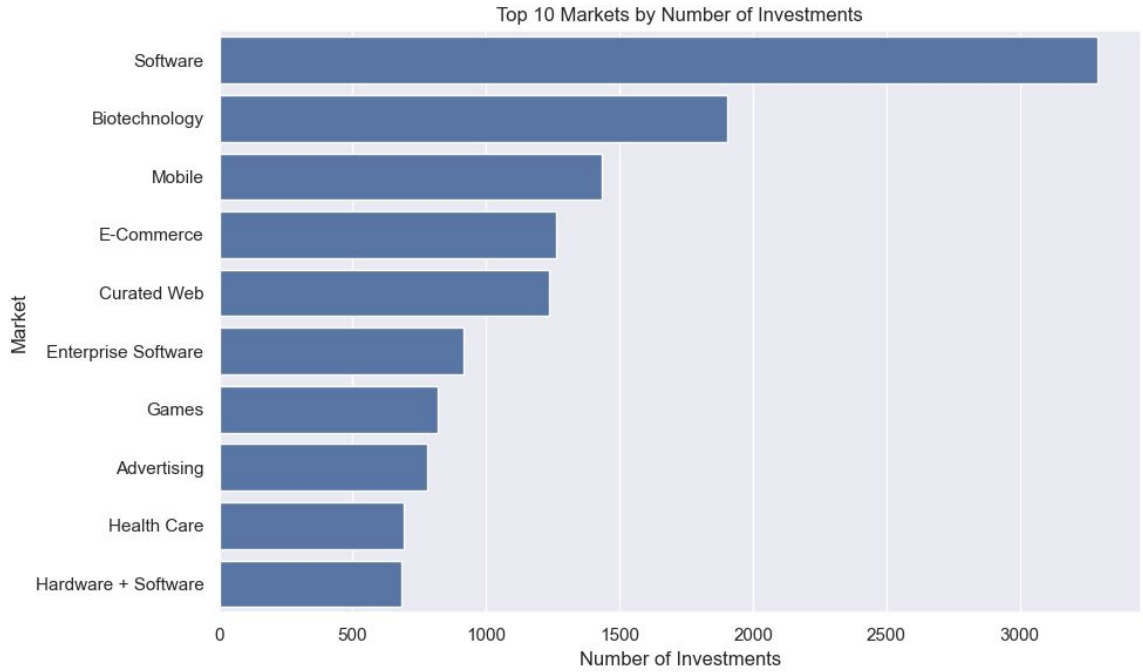
4.Correlation

- Country-wise Funding Insights



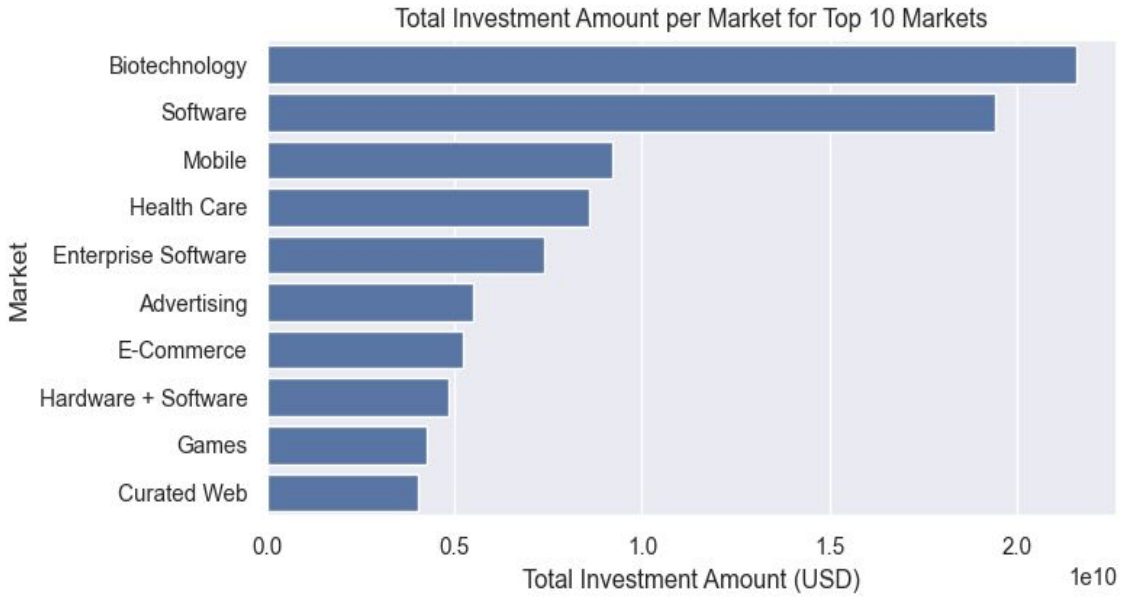
4.Correlation

- Markets' popular Investments



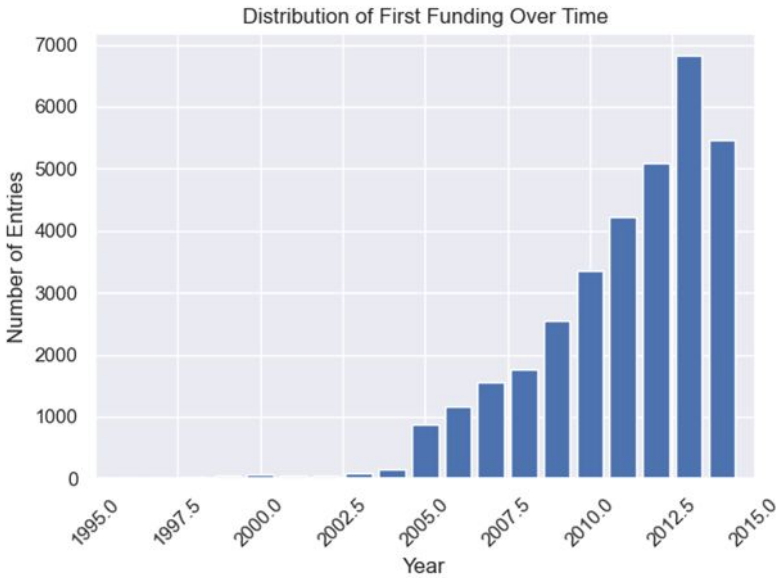
4.Correlation

- Market with investment Amound

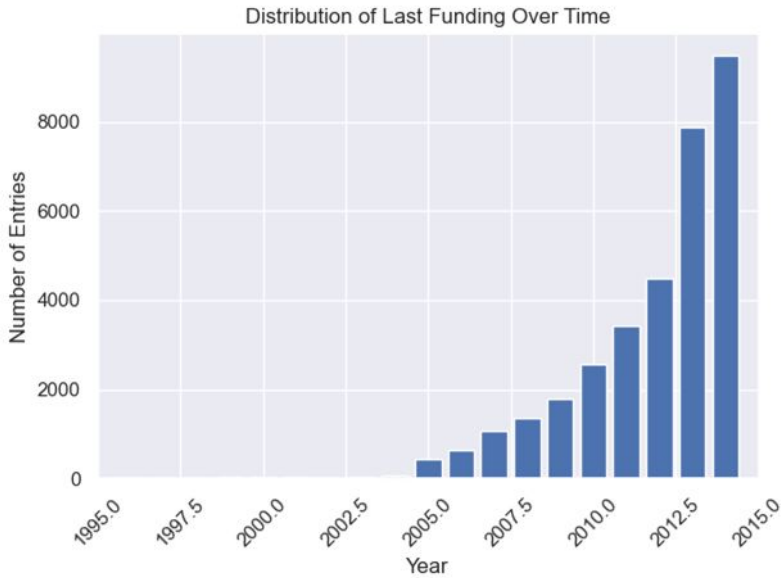


4.Correlation

- Distribution of First Funding Year

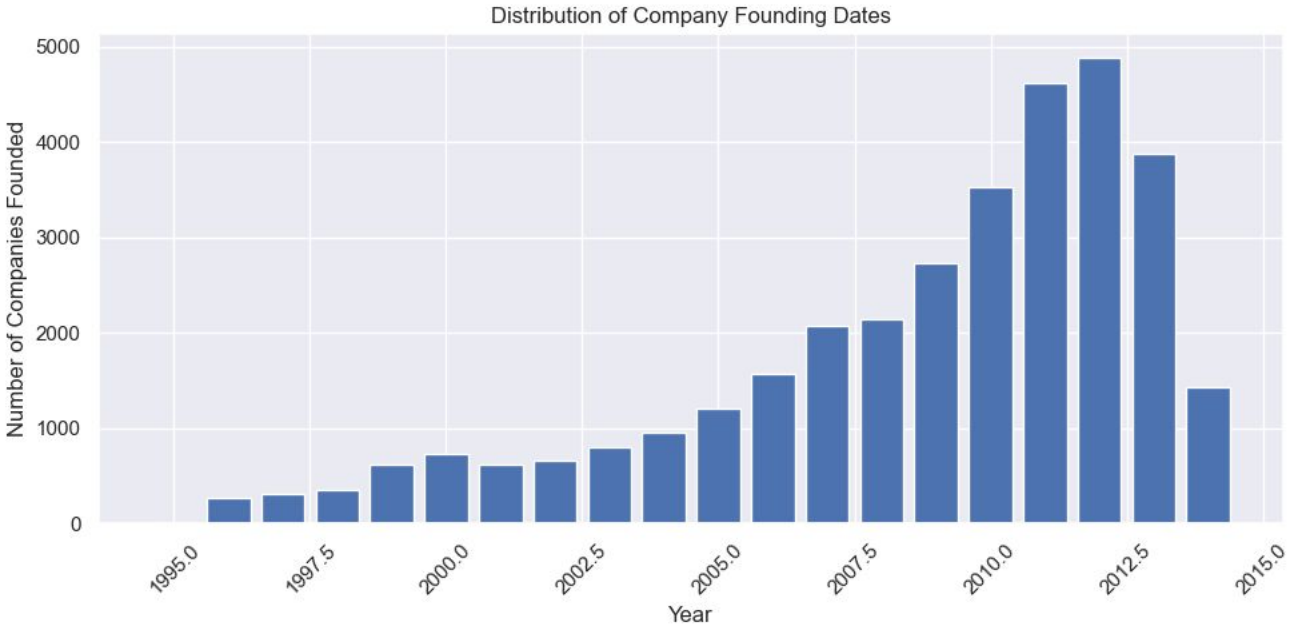


- Distribution of Last Funding Year



4.Correlation

- Distribution of Company Founding Year




Thank You