Predicting Startup Success - Machine Learning II Project

By: Katleen McQuiddy, Lindsay Neff, Stephanie Palanca, Prachi Pathak, and Lejla Skahic

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression, Ridge, RidgeCV, Lasso, LassoCV
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import neighbors
from sklearn.model_selection import KFold, cross_val_score
```

In [2]:
```
pip install eli5
```

```
Requirement already satisfied: eli5 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (0.13.0)
Requirement already satisfied: jinja2>=3.0.0 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5) (3.
1.2)
Requirement already satisfied: scikit-learn>=0.20 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5)
(1.0.2)
Requirement already satisfied: six in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5) (1.16.0)
Requirement already satisfied: numpy>=1.9.0 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5) (1.2
1.5)
Requirement already satisfied: scipy in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5) (1.7.3)
Requirement already satisfied: graphviz in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5) (0.20.1)
Requirement already satisfied: tabulate>=0.7.7 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5)
(0.8.9)
Requirement already satisfied: attrs>17.1.0 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from eli5) (21.
4.0)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from jinja2>=
3.0.0->eli5) (2.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from sci
kit-learn>=0.20->eli5) (2.2.0)
Requirement already satisfied: joblib>=0.11 in /Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages (from scikit-lear
n>=0.20->eli5) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import eli5
```

In [185]:
```
# allow access to google drive files - if using google drive
from google.colab import drive
drive.mount('/content/drive')
```

In [186]:
```
#change directory to project files page - if using google drive
import os
os.chdir('/content/drive/MyDrive/ML2_Project')
```

```
[CV] END bootstrap=False, max_depth=None, max_features=log2, min_samples_leaf=2, min_samples_split=2, n_estimators=1155; total tim
e=   1.5s
[CV] END bootstrap=False, max_depth=100, max_features=None, min_samples_leaf=4, min_samples_split=10, n_estimators=733; total time
=   3.4s
[CV] END bootstrap=False, max_depth=20, max_features=None, min_samples_leaf=1, min_samples_split=2, n_estimators=1366; total time=
8.2s
[CV] END bootstrap=True, max_depth=50, max_features=log2, min_samples_leaf=2, min_samples_split=2, n_estimators=944; total time=
1.3s
[CV] END bootstrap=True, max_depth=70, max_features=None, min_samples_leaf=4, min_samples_split=2, n_estimators=2000; total time=
7.0s
[CV] END bootstrap=False, max_depth=70, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=944; total time=
1.4s
[CV] END bootstrap=False, max_depth=80, max_features=None, min_samples_leaf=1, min_samples_split=10, n_estimators=100; total time=
0.6s
[CV] END bootstrap=True, max_depth=70, max_features=None, min_samples_leaf=2, min_samples_split=2, n_estimators=522; total time=
2.0s
[CV] END bootstrap=False, max_depth=40, max_features=None, min_samples_leaf=4, min_samples_split=2, n_estimators=1788; total time=
8.2s
[CV] END bootstrap=True, max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=2, n_estimators=1155; total time
```

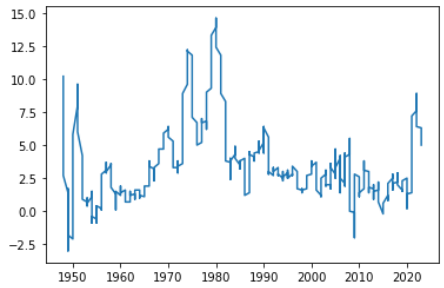
Economic Data

In [5]:
```
econ_data = pd.read_csv("UnempCPI2.csv")
econ_data.head()
```

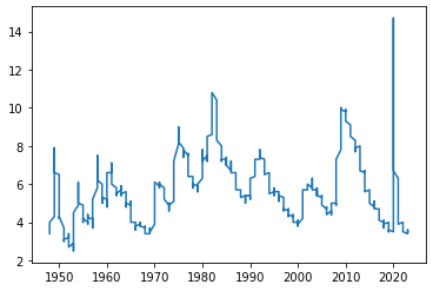
Out[5]:

	Year	Month	CPI	Unemployment
0	1948	1	10.2	3.4
1	1948	2	9.5	3.8
2	1948	3	6.8	4.0
3	1948	4	8.3	3.9
4	1948	5	9.4	3.5

In [6]:
```
plt.plot(econ_data['Year'],econ_data['CPI'])  # Plot the chart
plt.show()
```



In [7]:
```
plt.plot(econ_data['Year'],econ_data['Unemployment'])  # Plot the chart
plt.show()
```



In [8]:
```
econ_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 903 entries, 0 to 902
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Year          903 non-null    int64  
 1   Month         903 non-null    int64  
 2   CPI           903 non-null    float64
 3   Unemployment  903 non-null    float64
dtypes: float64(2), int64(2)
memory usage: 28.3 KB
```

Startup Data

Data Cleansing - Startup Data

In [9]:
```
df_startup = pd.read_csv("startup_data_final1.csv")
df_startup.head()
```

Out[9]:

	Organization Name	Industries	Headquarters Location	Founded Date	Operating Status	Closed Date	Number of Articles	Investor Type	Investment Stage	Number of Portfolio Organizations	...	Number of Contacts	SEMrush - Monthly Visits	SEMrush - Average Visits (6 months)	Active Tech Count	Numb of Ap
0	Kluster	Analytics, Artificial Intelligence, Machine Le...	London, England, United Kingdom	8/1/16	Active	NaN	3	NaN	NaN	NaN	...	NaN	3,629	4,494	6.0	
1	Credit Karma	Credit, Finance, Financial Services, FinTech, ...	San Francisco, California, United States	1/1/07	Active	NaN	661	NaN	NaN	1.0	...	1,037	80,947,049	72,541,070.83	70.0	
2	ecoATM	Consumer Electronics, Recycling, Waste Management	San Diego, California, United States	8/2/08	Active	NaN	121	NaN	NaN	NaN	...	150	1,071,721	1,602,595.17	41.0	
3	Reddit	Content, News, Social Media, Social Network	San Francisco, California, United States	1/1/05	Active	NaN	6,834	NaN	NaN	1.0	...	189	5,409,446,308	4,959,692,115.83	7.0	
4	Canopy	Artificial Intelligence, Information Services,...	Detroit, Michigan, United States	1/1/22	Active	NaN	81	NaN	NaN	NaN	...	25	70,818	87,030.33	54.0	N

5 rows × 49 columns

In [10]:
```
df_startup.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1111 entries, 0 to 1110
Data columns (total 49 columns):
 #   Column                                        Non-Null Count  Dtype  
---  ------                                        --------------  -----  
 0   Organization Name                             1111 non-null   object 
 1   Industries                                    1110 non-null   object 
 2   Headquarters Location                         1110 non-null   object 
 3   Founded Date                                  1108 non-null   object 
 4   Operating Status                              1111 non-null   object 
 5   Closed Date                                   237 non-null    object 
 6   Number of Articles                            930 non-null    object 
 7   Investor Type                                 1 non-null      object 
 8   Investment Stage                              1 non-null      object 
 9   Number of Portfolio Organizations             23 non-null     float64
 10  Number of Investments                         23 non-null     float64
 11  Number of Lead Investments                    9 non-null      float64
 12  Number of Exits                               8 non-null      float64
 13  Number of Exits (IPO)                         8 non-null      float64
 14  Industry Groups                               1110 non-null   object 
 15  Number of Founders                            875 non-null    float64
 16  Number of Employees                           1091 non-null   object 
 17  Number of Funding Rounds                      1094 non-null   float64
 18  Funding Status                                993 non-null    object 
 19  Last Funding Date                             1094 non-null   object 
 20  Last Funding Amount Currency (in USD)         1027 non-null   float64
 21  Last Funding Type                             1094 non-null   object 
 22  Last Equity Funding Amount Currency (in USD)  1027 non-null   float64
 23  Last Equity Funding Type                      1089 non-null   object 
 24  Total Equity Funding Amount Currency (in USD) 1081 non-null   float64
 25  Total Funding Amount Currency (in USD)        1086 non-null   float64
 26  Number of Lead Investors                      790 non-null    float64
 27  Number of Investors                           1072 non-null   float64
 28  Number of Acquisitions                        126 non-null    float64
 29  Acquisition Status                            717 non-null    object 
 30  Acquired by                                   694 non-null    object 
 31  Price Currency (in USD)                       215 non-null    float64
 32  Acquisition Type                              464 non-null    object 
 33  IPO Status                                    1111 non-null   object 
 34  IPO Date                                      13 non-null     object 
 35  Money Raised at IPO Currency (in USD)         11 non-null     float64
 36  Valuation at IPO                              1 non-null      float64
 37  Valuation at IPO Currency (in USD)            1 non-null      float64
 38  Number of Events                              94 non-null     float64
 39  Number of Contacts                            278 non-null    object 
 40  SEMrush - Monthly Visits                      230 non-null    object 
 41  SEMrush - Average Visits (6 months)           159 non-null    object 
 42  Active Tech Count                             1056 non-null   float64
 43  Number of Apps                                170 non-null    float64
 44  Total Products Active                         780 non-null    float64
 45  Patents Granted                               278 non-null    float64
 46  Trademarks Registered                         278 non-null    float64
 47   IT Spend Currency (in USD)                   35 non-null     float64
 48  Estimated Revenue Range                       272 non-null    object 
dtypes: float64(25), object(24)
memory usage: 425.4+ KB
```

In [11]:
```
df_startup = df_startup.drop(columns=['Investor Type', 'Investment Stage', 'Number of Portfolio Organizations', 'Number of Investment
```

In [12]:
```
df_startup['Founded Date'] = pd.to_datetime(df_startup['Founded Date'])
df_startup['Closed Date'] = pd.to_datetime(df_startup['Closed Date'])
df_startup['Last Funding Date'] = pd.to_datetime(df_startup['Last Funding Date'])
```

In [13]:
```
df_startup['Founded Year'] = df_startup['Founded Date'].dt.year
df_startup['Founded Month'] = df_startup['Founded Date'].dt.month
df_startup['Closed Year'] = df_startup['Closed Date'].dt.year
df_startup['Closed Month'] = df_startup['Closed Date'].dt.month
```

In [14]:
```
float_cols = df_startup.select_dtypes(include=['float']).columns
df_startup[float_cols] = df_startup[float_cols].fillna(0)
```

In [15]:
```
obj_cols = df_startup.select_dtypes(include=['object']).columns
df_startup[obj_cols] = df_startup[obj_cols].fillna('N/A')
```

In [16]:
```
df_startup.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1111 entries, 0 to 1110
Data columns (total 34 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Organization Name                             1111 non-null   object
 1   Industries                                    1111 non-null   object
 2   Headquarters Location                         1111 non-null   object
 3   Founded Date                                  1108 non-null   datetime64[ns]
 4   Operating Status                              1111 non-null   object
 5   Closed Date                                   237 non-null    datetime64[ns]
 6   Number of Articles                            1111 non-null   object
 7   Industry Groups                               1111 non-null   object
 8   Number of Founders                            1111 non-null   float64
 9   Number of Employees                           1111 non-null   object
 10  Number of Funding Rounds                      1111 non-null   float64
 11  Funding Status                                1111 non-null   object
 12  Last Funding Date                             1094 non-null   datetime64[ns]
 13  Last Funding Amount Currency (in USD)         1111 non-null   float64
 14  Last Funding Type                             1111 non-null   object
 15  Last Equity Funding Amount Currency (in USD)  1111 non-null   float64
 16  Last Equity Funding Type                      1111 non-null   object
 17  Total Equity Funding Amount Currency (in USD) 1111 non-null   float64
 18  Total Funding Amount Currency (in USD)        1111 non-null   float64
 19  Number of Investors                           1111 non-null   float64
 20  Acquisition Status                            1111 non-null   object
 21  Acquired by                                   1111 non-null   object
 22  Price Currency (in USD)                       1111 non-null   float64
 23  IPO Status                                    1111 non-null   object
 24  Active Tech Count                             1111 non-null   float64
 25  Number of Apps                                1111 non-null   float64
 26  Total Products Active                         1111 non-null   float64
 27  Patents Granted                               1111 non-null   float64
 28  Trademarks Registered                         1111 non-null   float64
 29  Estimated Revenue Range                       1111 non-null   object
 30  Founded Year                                  1111 non-null   float64
 31  Founded Month                                 1111 non-null   float64
 32  Closed Year                                   1111 non-null   float64
 33  Closed Month                                  1111 non-null   float64
dtypes: datetime64[ns](3), float64(17), object(14)
memory usage: 295.2+ KB
```

Exploratory Data Analysis

In [17]:
```
df_startup.describe()
```

Out[17]:

	Number of Founders	Number of Funding Rounds	Last Funding Amount Currency (in USD)	Last Equity Funding Amount Currency (in USD)	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	Active Tech Count	Number of Apps	Total Products Active	Patents Granted	Trade Regi
count	1111.000000	1111.000000	1.111000e+03	1.111000e+03	1.111000e+03	1.111000e+03	1111.000000	1.111000e+03	1111.000000	1111.000000	1111.000000	1111.000000	1111.0
mean	1.541854	3.015302	8.912437e+06	8.906442e+06	2.947140e+07	3.264480e+07	4.651665	5.914462e+07	13.819082	0.988299	7.508551	2.701170	0.9
std	1.181617	1.976606	2.391056e+07	1.813687e+07	1.828948e+08	2.126899e+08	3.954265	3.114099e+08	19.322270	7.925489	11.180004	22.604249	6.3
min	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.0
25%	1.000000	2.000000	1.000000e+06	1.000000e+06	2.600000e+06	2.800000e+06	2.000000	0.000000e+00	3.000000	0.000000	0.000000	0.000000	0.0
50%	1.000000	3.000000	4.000000e+06	4.256180e+06	1.050000e+07	1.125308e+07	4.000000	0.000000e+00	7.000000	0.000000	4.000000	0.000000	0.0
75%	2.000000	4.000000	1.000000e+07	1.000000e+07	3.015000e+07	3.145500e+07	6.000000	0.000000e+00	17.000000	0.000000	10.000000	0.000000	0.0
max	6.000000	17.000000	5.000000e+08	4.100000e+08	5.864000e+09	6.784000e+09	35.000000	7.100000e+09	224.000000	229.000000	125.000000	584.000000	97.0

In [18]:
```
df_startup['Estimated Revenue Range'].unique()
```

Out[18]:
```
array(['Less than $1M', '$100M to $500M', '$50M to $100M', 'N/A',
       '$1M to $10M', '$10B+', '$10M to $50M', '$1B to $10B',
       '$500M to $1B'], dtype=object)
```

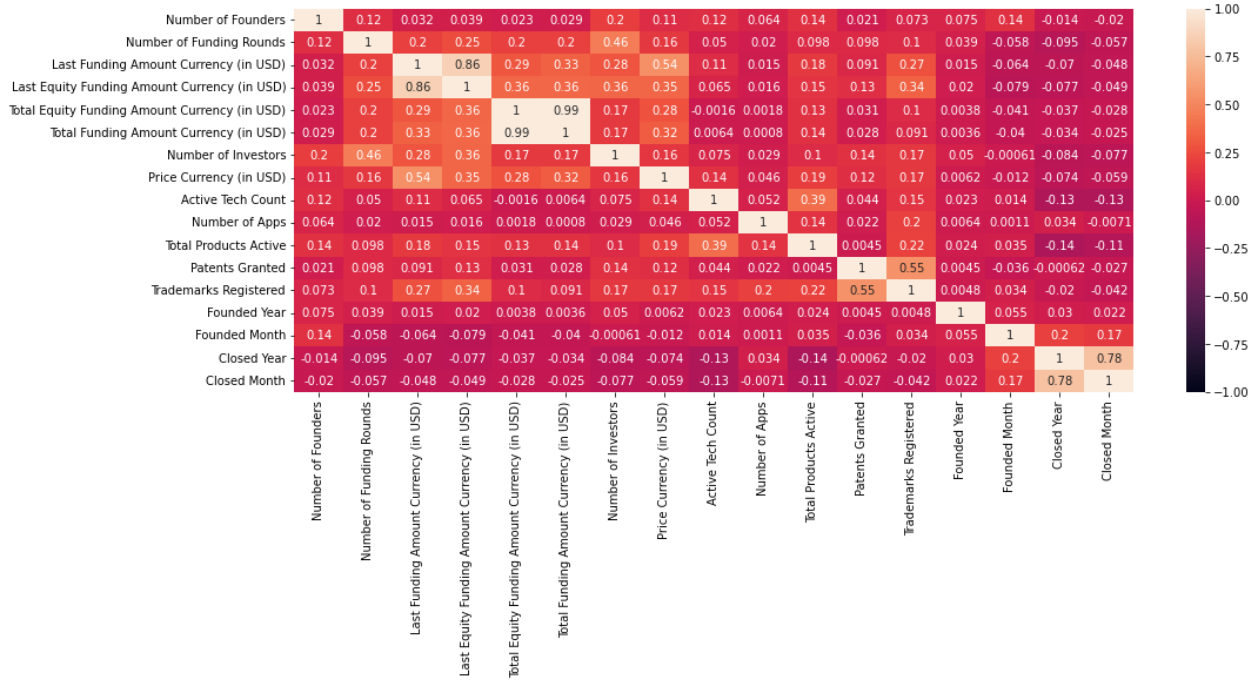
In [19]:
```
df_startup.loc[df_startup['Total Funding Amount Currency (in USD)'] > 500000000,['Total Funding Amount Currency (in USD)']]
```

Out[19]:

	Total Funding Amount Currency (in USD)
1	8.680000e+08
3	1.329050e+09
139	6.784000e+09
303	6.170000e+08
537	6.930000e+08

In [20]:
```
plt.figure(figsize=(16,6))
sns.heatmap(df_startup.corr(), vmin=-1, vmax=1, annot=True)

#number of investors and number of funding rounds
#trademarks registered and last equity funding amount currency
#active tech count and total products active
#patents active and trademarks registered
#patents active and number of investors
#Number of founders and number of investors
```

Out[20]: <AxesSubplot:>



In [21]:
```
df_startup['Headquarters Location'].unique()
```

Out[21]:
```
array(['London, England, United Kingdom',
       'San Francisco, California, United States',
       'San Diego, California, United States',
       'Detroit, Michigan, United States',
       'Carlsbad, California, United States',
       'San Bruno, California, United States', 'Berlin, Berlin, Germany',
       'Miami, Florida, United States',
       'Chicago, Illinois, United States',
       'South Melbourne, Victoria, Australia',
       'Mill Valley, California, United States',
       'Mumbai, Maharashtra, India', 'Marietta, Georgia, United States',
       'Edmond, Oklahoma, United States',
       'Oakland, California, United States',
       'Austin, Texas, United States',
       'New York, New York, United States',
       'Sunnyvale, California, United States',
       'Plano, Texas, United States',
       'Mountain View, California, United States',
       'Tucson, Arizona, United States', 'Tel Aviv, Tel Aviv, Israel',
```

Importing a dataset with the organization name, industry, and state

In [22]:
```
df_industry = pd.read_csv('industry.csv')
df_industry.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1154 entries, 0 to 1153
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   name           1154 non-null   object
 1   state_code     1154 non-null   object
 2   category_code  1154 non-null   object
dtypes: object(3)
memory usage: 27.2+ KB
```

In [23]:
```
df_industry = df_industry.rename(columns={'name': 'Organization Name'})
```

Merging the CrunchBase Dataset with the Industry Dataset

In [24]:
```
df_startup2 = pd.merge(left=df_startup, right=df_industry, on='Organization Name', how="inner")
df_startup2
```

Out[24]:

	Organization Name	Industries	Headquarters Location	Founded Date	Operating Status	Closed Date	Number of Articles	Industry Groups	Number of Founders	Number of Employees	...	Total Products Active	Patents Granted	Trademarks Registered	Estimated Revenue Range	Fou
0	Kluster	Analytics, Artificial Intelligence, Machine Le...	London, England, United Kingdom	2016-08-01	Active	NaT	3	Artificial Intelligence, Data and Analytics, I...	2.0	Nov-50	...	29.0	0.0	0.0	Less than $1M	2
1	ecoATM	Consumer Electronics, Recycling, Waste Management	San Diego, California, United States	2008-08-02	Active	NaT	121	Consumer Electronics, Hardware, Sustainability	3.0	501-1000	...	36.0	63.0	5.0	$50M to 100M	2
2	Reddit	Content, News, Social Media, Social Network	San Francisco, California, United States	2005-01-01	Active	NaT	6,834	Content and Publishing, Internet Services, Med...	3.0	501-1000	...	63.0	0.0	78.0	$100M to 500M	2
3	Aptera	Automotive, Manufacturing, Service Industry	Carlsbad, California, United States	2006-01-01	Closed	NaT	109	Manufacturing, Other, Transportation	2.0	51-100	...	0.0	0.0	0.0	$1M to 10M	2
4	Kyte	Automotive, Fleet Management, Rental, Software...	San Francisco, California, United States	2019-01-01	Active	NaT	5	Commerce and Shopping, Software, Transportation	3.0	101-250	...	0.0	0.0	1.0	$1M to 10M	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1054	Dispatch	Delivery, Information Technology, Software	Bloomington, Minnesota, United States	2016-08-01	Active	NaT	6	Administrative Services, Information Technolog...	2.0	101-250	...	25.0	0.0	0.0	$1M to 10M	2
1055	Pulse	Crowdsourcing, IT Management, Market Research,...	San Francisco, California, United States	2017-04-01	Active	NaT	6	Community and Lifestyle, Data and Analytics, D...	2.0	10-Jan	...	34.0	0.0	1.0	$1M to 10M	2
1056	Wize	EdTech	Vancouver, British Columbia, Canada	2017-07-01	Active	NaT	4	Education, Software	3.0	Nov-50	...	18.0	0.0	0.0	$10M to 50M	2
1057	Astrid	E-Learning, EdTech, Education	Stockholm, Stockholms Lan, Sweden	2020-01-01	Active	NaT	N/A	Education, Software	2.0	10-Jan	...	0.0	0.0	0.0	N/A	2
1058	Lore	Blockchain, Cryptocurrency, Ethereum, Social N...	San Francisco, California, United States	2020-05-01	Active	NaT	N/A	Financial Services, Internet Services, Other, ...	2.0	10-Jan	...	0.0	0.0	0.0	N/A	2

1059 rows × 36 columns

In [25]:
```
df_startup2.info()
```

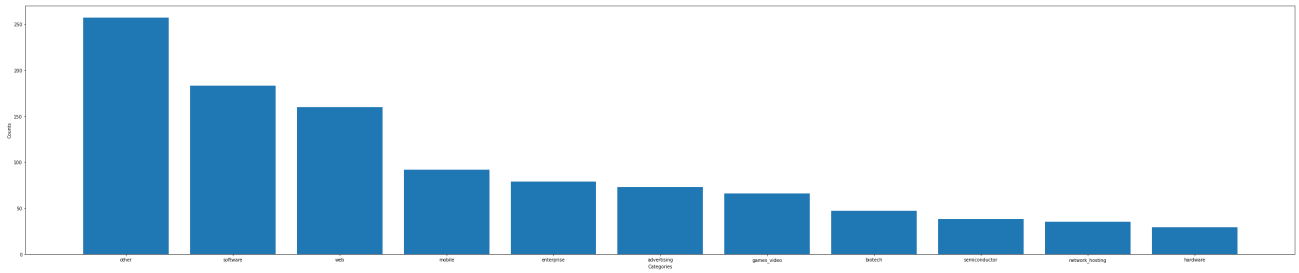
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1059 entries, 0 to 1058
Data columns (total 36 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Organization Name                             1059 non-null   object
 1   Industries                                    1059 non-null   object
 2   Headquarters Location                         1059 non-null   object
 3   Founded Date                                  1057 non-null   datetime64[ns]
 4   Operating Status                              1059 non-null   object
 5   Closed Date                                   231 non-null    datetime64[ns]
 6   Number of Articles                            1059 non-null   object
 7   Industry Groups                               1059 non-null   object
 8   Number of Founders                            1059 non-null   float64
 9   Number of Employees                           1059 non-null   object
 10  Number of Funding Rounds                      1059 non-null   float64
 11  Funding Status                                1059 non-null   object
 12  Last Funding Date                             1052 non-null   datetime64[ns]
 13  Last Funding Amount Currency (in USD)         1059 non-null   float64
 14  Last Funding Type                             1059 non-null   object
 15  Last Equity Funding Amount Currency (in USD)  1059 non-null   float64
 16  Last Equity Funding Type                      1059 non-null   object
 17  Total Equity Funding Amount Currency (in USD) 1059 non-null   float64
 18  Total Funding Amount Currency (in USD)        1059 non-null   float64
 19  Number of Investors                           1059 non-null   float64
 20  Acquisition Status                            1059 non-null   object
 21  Acquired by                                   1059 non-null   object
 22  Price Currency (in USD)                       1059 non-null   float64
 23  IPO Status                                    1059 non-null   object
 24  Active Tech Count                             1059 non-null   float64
 25  Number of Apps                                1059 non-null   float64
 26  Total Products Active                         1059 non-null   float64
 27  Patents Granted                               1059 non-null   float64
 28  Trademarks Registered                         1059 non-null   float64
 29  Estimated Revenue Range                       1059 non-null   object
 30  Founded Year                                  1059 non-null   float64
 31  Founded Month                                 1059 non-null   float64
 32  Closed Year                                   1059 non-null   float64
 33  Closed Month                                  1059 non-null   float64
 34  state_code                                    1059 non-null   object
 35  category_code                                 1059 non-null   object
dtypes: datetime64[ns](3), float64(17), object(16)
memory usage: 306.1+ KB
```

Exporatory Data Analysis with the Merged Startup Data

In [26]:
```
# Get a list of the top 10 categories
top10 = df_startup2['category_code'].value_counts()[:10].index

# replace the category with 'OTHER' if not in top 10
df_startup2.loc[~df_startup2['category_code'].isin(top10), 'category_code'] = 'other'
```

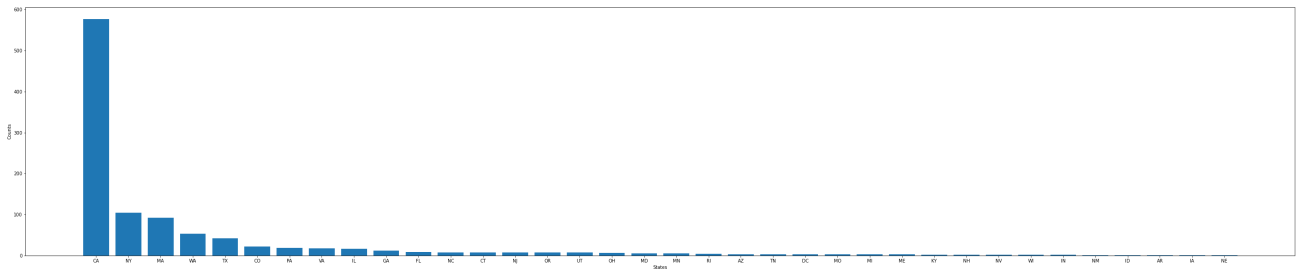
In [27]:
```
industry = df_startup2["category_code"].value_counts()
fig, ax = plt.subplots(figsize=(50, 10))
plt.bar(industry.index, industry.values)
plt.xlabel("Categories")
plt.ylabel("Counts")
```

Out[27]: Text(0, 0.5, 'Counts')



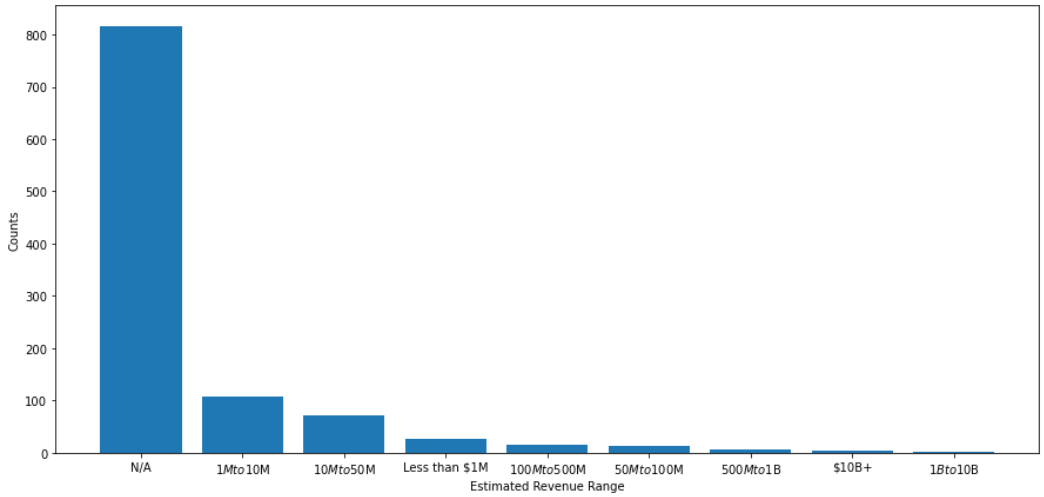
In [28]:
```
#Viewing number of states
state = df_startup2["state_code"].value_counts()
fig, ax = plt.subplots(figsize=(50, 10))
plt.bar(state.index, state.values)
plt.xlabel("States")
plt.ylabel("Counts")
```

Out[28]: Text(0, 0.5, 'Counts')



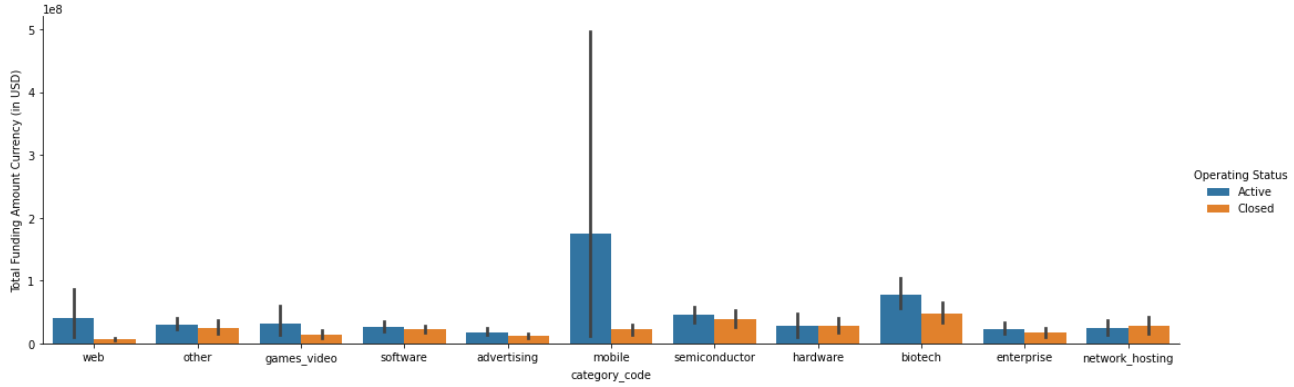
In [29]:
```
#getting funding counts
revenue = df_startup2["Estimated Revenue Range"].value_counts()
fig, ax = plt.subplots(figsize=(15, 7))
plt.bar(revenue.index.astype(str), revenue.values)
plt.xlabel("Estimated Revenue Range")
plt.ylabel("Counts")
```

Out[29]: Text(0, 0.5, 'Counts')



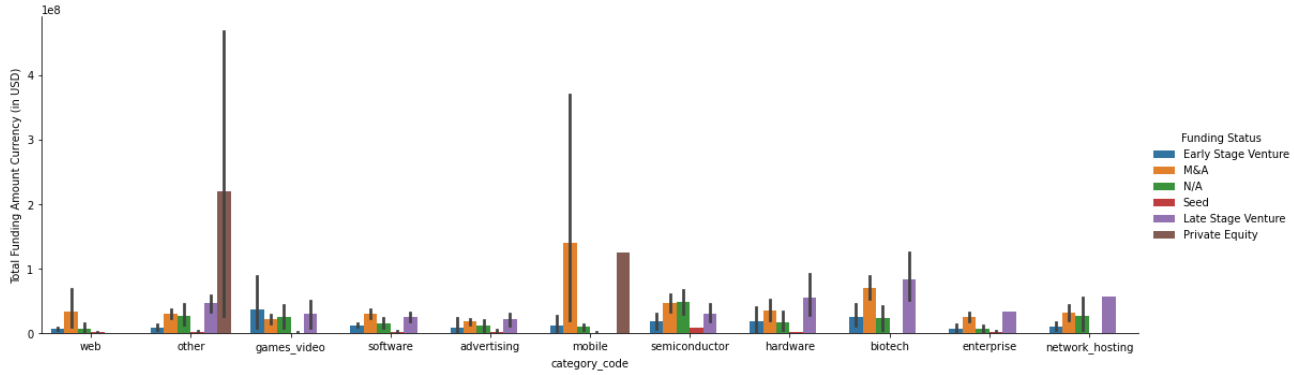
In [30]:
```
sns.catplot(x = "category_code",       # x variable name
            y = "Total Funding Amount Currency (in USD)",       # y variable name
            hue = "Operating Status",  # group variable name
            data = df_startup2,     # dataframe to plot
            kind = "bar",
            height = 5, aspect = 3)
```

Out[30]: <seaborn.axisgrid.FacetGrid at 0x7ffcf86ab3a0>



In [31]:
```
sns.catplot(x = "category_code",       # x variable name
            y = "Total Funding Amount Currency (in USD)",       # y variable name
            hue = "Funding Status",  # group variable name
            data = df_startup2,     # dataframe to plot
            kind = "bar",
            height = 5, aspect = 3)
```

Out[31]: <seaborn.axisgrid.FacetGrid at 0x7ffce9361310>



In [32]:
```
df_startup2['Founded Year'].unique()
```

Out[32]:
```
array([2016., 2008., 2005., 2006., 2019., 2015., 2014., 2020., 2004.,
       2017., 2021., 2009., 2018., 2007., 2022., 2010., 2013., 2000.,
          0., 2001., 1996., 2003., 2002., 2011., 1999., 1998., 1997.,
       2012., 1985., 1990., 1995., 1984., 1992., 1978.])
```

In [33]:
```
#Changing Founded Year to Year so we are able to match the years of the startup data and the economic data
df_startup2 = df_startup2.rename(columns={'Founded Year': 'Year'})
```

In [34]:
```
#Changing Founded Month to Month so we are able to match the months of the startup data with the economic data
df_startup2 = df_startup2.rename(columns={'Founded Month': 'Month'})
```

Merging the Startup Data with the Economic Data

In [35]:
```
merged_df_test= pd.merge(left=df_startup2, right=econ_data, on =['Year', 'Month'], how ='left')
merged_df_test.head()
```

Out[35]:

	Organization Name	Industries	Headquarters Location	Founded Date	Operating Status	Closed Date	Number of Articles	Industry Groups	Number of Founders	Number of Employees	...	Trademarks Registered	Estimated Revenue Range	Year	Month	Closed Year	Clo Mc
0	Kluster	Analytics, Artificial Intelligence, Machine Le...	London, England, United Kingdom	2016-08-01	Active	NaT	3	Artificial Intelligence, Data and Analytics, I...	2.0	Nov-50	...	0.0	Less than $1M	2016.0	8.0	0.0	
1	ecoATM	Consumer Electronics, Recycling, Waste Management	San Diego, California, United States	2008-08-02	Active	NaT	121	Consumer Electronics, Hardware, Sustainability	3.0	501-1000	...	5.0	$50M to 100M	2008.0	8.0	0.0	
2	Reddit	Content, News, Social Media, Social Network	San Francisco, California, United States	2005-01-01	Active	NaT	6,834	Content and Publishing, Internet Services, Med...	3.0	501-1000	...	78.0	$100M to 500M	2005.0	1.0	0.0	
3	Aptera	Automotive, Manufacturing, Service Industry	Carlsbad, California, United States	2006-01-01	Closed	NaT	109	Manufacturing, Other, Transportation	2.0	51-100	...	0.0	$1M to 10M	2006.0	1.0	0.0	
4	Kyte	Automotive, Fleet Management, Rental, Software...	San Francisco, California, United States	2019-01-01	Active	NaT	5	Commerce and Shopping, Software, Transportation	3.0	101-250	...	1.0	$1M to 10M	2019.0	1.0	0.0	

5 rows × 38 columns

In [36]:
```
merged_df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1059 entries, 0 to 1058
Data columns (total 38 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Organization Name                             1059 non-null   object
 1   Industries                                    1059 non-null   object
 2   Headquarters Location                         1059 non-null   object
 3   Founded Date                                  1057 non-null   datetime64[ns]
 4   Operating Status                              1059 non-null   object
 5   Closed Date                                   231 non-null    datetime64[ns]
 6   Number of Articles                            1059 non-null   object
 7   Industry Groups                               1059 non-null   object
 8   Number of Founders                            1059 non-null   float64
 9   Number of Employees                           1059 non-null   object
 10  Number of Funding Rounds                      1059 non-null   float64
 11  Funding Status                                1059 non-null   object
 12  Last Funding Date                             1052 non-null   datetime64[ns]
 13  Last Funding Amount Currency (in USD)         1059 non-null   float64
 14  Last Funding Type                             1059 non-null   object
 15  Last Equity Funding Amount Currency (in USD)  1059 non-null   float64
 16  Last Equity Funding Type                      1059 non-null   object
 17  Total Equity Funding Amount Currency (in USD) 1059 non-null   float64
 18  Total Funding Amount Currency (in USD)        1059 non-null   float64
 19  Number of Investors                           1059 non-null   float64
 20  Acquisition Status                            1059 non-null   object
 21  Acquired by                                   1059 non-null   object
 22  Price Currency (in USD)                       1059 non-null   float64
 23  IPO Status                                    1059 non-null   object
 24  Active Tech Count                             1059 non-null   float64
 25  Number of Apps                                1059 non-null   float64
 26  Total Products Active                         1059 non-null   float64
 27  Patents Granted                               1059 non-null   float64
 28  Trademarks Registered                         1059 non-null   float64
 29  Estimated Revenue Range                       1059 non-null   object
 30  Year                                          1059 non-null   float64
 31  Month                                         1059 non-null   float64
 32  Closed Year                                   1059 non-null   float64
 33  Closed Month                                  1059 non-null   float64
 34  state_code                                    1059 non-null   object
 35  category_code                                 1059 non-null   object
 36  CPI                                           1057 non-null   float64
 37  Unemployment                                  1057 non-null   float64
dtypes: datetime64[ns](3), float64(19), object(16)
memory usage: 322.7+ KB
```

Data Wrangling with the Merged Data

In [37]:
```
merged_df_test['Number of Employees'].unique()
```

Out[37]:
```
array(['Nov-50', '501-1000', '51-100', '101-250', '1001-5000', '10-Jan',
       '5001-10000', '251-500', '10001+', 'N/A'], dtype=object)
```

In [38]:
```
#Replacing the values presented as dates with their respective # of employee ranges
merged_df_test['Number of Employees'].replace('Nov-50', '11-50', inplace=True)
merged_df_test['Number of Employees'].replace('10-Jan', '1-10', inplace=True)
```

In [39]:
```
merged_df_test['Acquisition Status'].unique()
```

Out[39]:
```
array(['N/A', 'Was Acquired', 'Made Acquisitions, Was Acquired',
       'Made Acquisitions'], dtype=object)
```

In [40]:
```
#Creating a new Acquisition Status so it only contains binary variables
merged_df_test['Acquisition'] = merged_df_test['Acquisition Status'].apply(lambda x: 'Acquired' if ('Was Acquired' in x) or ('Made Ac
```

In [41]:
```
#Dropping the Founded Data, Closed Date and Last Funding Dates, since we no longer need them
merged_df_test = merged_df_test.drop(columns=['Founded Date','Closed Date','Last Funding Date'])
```

In [42]:
```
print(merged_df_test.isna().sum())
```

```
Organization Name                              0
Industries                                     0
Headquarters Location                          0
Operating Status                               0
Number of Articles                             0
Industry Groups                                0
Number of Founders                             0
Number of Employees                            0
Number of Funding Rounds                       0
Funding Status                                 0
Last Funding Amount Currency (in USD)          0
Last Funding Type                              0
Last Equity Funding Amount Currency (in USD)   0
Last Equity Funding Type                       0
Total Equity Funding Amount Currency (in USD)  0
Total Funding Amount Currency (in USD)         0
Number of Investors                            0
Acquisition Status                             0
Acquired by                                    0
Price Currency (in USD)                        0
IPO Status                                     0
Active Tech Count                              0
Number of Apps                                 0
Total Products Active                          0
Patents Granted                                0
Trademarks Registered                          0
Estimated Revenue Range                        0
Year                                           0
Month                                          0
Closed Year                                    0
Closed Month                                   0
state_code                                     0
category_code                                  0
CPI                                            2
Unemployment                                   2
Acquisition                                    0
dtype: int64
```

In [43]:
```
#Dropping the rows with missing data
merged_df_test = merged_df_test.dropna()
```

In [44]:
```
#Changing the name of the month and year column back to founded month and founded year
merged_df_test = merged_df_test.rename(columns={'Month': 'Founded Month'})
merged_df_test = merged_df_test.rename(columns={'Year': 'Founded Year'})

#Changing the names of the states and industry columns
merged_df_test = merged_df_test.rename(columns={'state_code': 'State'})
merged_df_test = merged_df_test.rename(columns={'category_code': 'Industry'})
```

In [45]:
```
#Replacing "N/A" with 0 and converting the Number of Articles column to a float
merged_df_test['Number of Articles'].replace('N/A', '0', inplace=True)
merged_df_test['Number of Articles'] = merged_df_test['Number of Articles'].str.replace(',', '').astype(float)
```

In [46]:
```
#Converting Founded Month to an object
merged_df_test['Founded Month'] = merged_df_test['Founded Month'].astype(object)
```

In [47]:
```
#Dropping columns we no longer need from the merged dataset/columns that are redundant
merged_df_test = merged_df_test.drop(columns=['Organization Name','Industries','Headquarters Location','Acquired by', 'Industry Grou
```

In [48]:
```
merged_df_test.describe()
```

Out[48]:

	Number of Funding Rounds	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	Total Products Active	Patents Granted	Trademarks Registered	Founded Year	Closed Year	Closed Month	CPI	Unemplo
count	1057.000000	1.057000e+03	1.057000e+03	1057.000000	1.057000e+03	1057.000000	1057.000000	1057.000000	1057.000000	1057.000000	1057.000000	1057.000000	1057.(
mean	3.030274	2.925149e+07	3.201847e+07	4.649007	5.390982e+07	7.219489	2.079470	0.763482	2005.771050	439.630085	1.145695	2.560360	5.8
std	1.933318	1.868314e+08	2.161818e+08	3.800765	2.322759e+08	10.930006	14.141912	5.908301	4.354834	831.720170	2.769826	1.381722	1.7
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000	0.000000	1978.000000	0.000000	0.000000	-2.000000	3.5
25%	2.000000	2.850000e+06	3.000000e+06	2.000000	0.000000e+00	0.000000	0.000000	0.000000	2003.000000	0.000000	0.000000	2.000000	4.6
50%	3.000000	1.080000e+07	1.145000e+07	4.000000	0.000000e+00	4.000000	0.000000	0.000000	2006.000000	0.000000	0.000000	2.800000	5.3
75%	4.000000	3.015000e+07	3.141000e+07	6.000000	0.000000e+00	10.000000	0.000000	0.000000	2008.000000	0.000000	0.000000	3.600000	5.8
max	17.000000	5.864000e+09	6.784000e+09	33.000000	4.000000e+09	125.000000	385.000000	97.000000	2022.000000	2022.000000	12.000000	8.200000	13.2

In [49]:
```
merged_df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1057 entries, 0 to 1058
Data columns (total 22 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   Number of Employees                         1057 non-null   object 
 1   Number of Funding Rounds                    1057 non-null   float64
 2   Last Funding Type                           1057 non-null   object 
 3   Last Equity Funding Type                    1057 non-null   object 
 4   Total Equity Funding Amount Currency (in USD)  1057 non-null   float64
 5   Total Funding Amount Currency (in USD)      1057 non-null   float64
 6   Number of Investors                         1057 non-null   float64
 7   Price Currency (in USD)                     1057 non-null   float64
 8   IPO Status                                  1057 non-null   object 
 9   Total Products Active                       1057 non-null   float64
 10  Patents Granted                             1057 non-null   float64
 11  Trademarks Registered                       1057 non-null   float64
 12  Estimated Revenue Range                     1057 non-null   object 
 13  Founded Year                                1057 non-null   float64
 14  Founded Month                               1057 non-null   object 
 15  Closed Year                                 1057 non-null   float64
 16  Closed Month                                1057 non-null   float64
 17  State                                       1057 non-null   object 
 18  Industry                                    1057 non-null   object 
 19  CPI                                         1057 non-null   float64
 20  Unemployment                                1057 non-null   float64
 21  Acquisition                                 1057 non-null   object 
dtypes: float64(13), object(9)
memory usage: 189.9+ KB
```

Pre Processing Steps

In [50]:
```
#Scaling our Funding Features, since those values are significantly larger than the other values
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer

num_cols = ['Total Equity Funding Amount Currency (in USD)',
            'Total Funding Amount Currency (in USD)',
            'Price Currency (in USD)']

scaled_num_df = merged_df_test[num_cols]

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num_cols),
    ])
transformed_data = preprocessor.fit_transform(merged_df_test)

scaled_num_cols = preprocessor.transformers_[0][2]

scaled_num_df = pd.DataFrame(transformed_data, columns=num_cols)
merged_df_test[scaled_num_cols] = scaled_num_df

scaled_mean = scaled_num_df.mean()
scaled_std = scaled_num_df.std()

print("Scaled mean:", scaled_mean)
print("Scaled std:", scaled_std)
```

```
Scaled mean: Total Equity Funding Amount Currency (in USD)    1.710500e-16
Total Funding Amount Currency (in USD)          -1.003087e-17
Price Currency (in USD)                          5.022262e-16
dtype: float64
Scaled std: Total Equity Funding Amount Currency (in USD)    1.000473
Total Funding Amount Currency (in USD)           1.000473
Price Currency (in USD)                          1.000473
dtype: float64
```

In [51]:
```
merged_df_test.describe()
```

Out[51]:

	Number of Funding Rounds	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	Total Products Active	Patents Granted	Trademarks Registered	Founded Year	Closed Year	Closed Month	CPI	Unemployme
count	1057.000000	1055.000000	1055.000000	1057.000000	1055.000000	1057.000000	1057.000000	1057.000000	1057.000000	1057.000000	1057.000000	1057.000000	1057.000(
mean	3.030274	-0.000055	-0.000024	4.649007	0.000440	7.219489	2.079470	0.763482	2005.771050	439.630085	1.145695	2.560360	5.816(
std	1.933318	1.001411	1.001414	3.800765	1.001371	10.930006	14.141912	5.908301	4.354834	831.720170	2.769826	1.381722	1.770(
min	0.000000	-0.156640	-0.148179	0.000000	-0.232204	0.000000	0.000000	0.000000	1978.000000	0.000000	0.000000	-2.000000	3.500(
25%	2.000000	-0.141513	-0.134295	2.000000	-0.232204	0.000000	0.000000	0.000000	2003.000000	0.000000	0.000000	2.000000	4.600(
50%	3.000000	-0.098807	-0.095560	4.000000	-0.232204	4.000000	0.000000	0.000000	2006.000000	0.000000	0.000000	2.800000	5.300(
75%	4.000000	0.004811	-0.003186	6.000000	-0.232204	10.000000	0.000000	0.000000	2008.000000	0.000000	0.000000	3.600000	5.800(
max	17.000000	31.244799	31.247675	33.000000	16.996851	125.000000	385.000000	97.000000	2022.000000	2022.000000	12.000000	8.200000	13.200(

In [52]:
```
merged_df_test = merged_df_test.dropna()
```

In [53]:
```
#Encoding the labels
from sklearn.preprocessing import LabelEncoder


encoder = LabelEncoder()
Y = encoder.fit_transform(merged_df_test['Acquisition'])
X = merged_df_test.drop(['Acquisition'], axis=1)

for col in X.select_dtypes(include='object').columns:
    X[col] = X[col].astype('category')

X = pd.get_dummies(data=X, drop_first=True)
```

In [54]:
```
X.head()
```

Out[54]:

	Number of Funding Rounds	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	Total Products Active	Patents Granted	Trademarks Registered	Founded Year	Closed Year	...	Industry_biotech	Industry_enterprise	Industry_games_video	Industry_
0	2.0	-0.124511	-0.120412	5.0	-0.232204	29.0	0.0	0.0	2016.0	0.0	...	0	0	0	
1	9.0	0.680874	1.693347	11.0	1.275339	36.0	63.0	5.0	2008.0	0.0	...	0	0	0	
2	10.0	6.960358	6.002566	33.0	-0.232204	63.0	0.0	78.0	2005.0	0.0	...	0	0	0	
3	9.0	0.370019	0.318547	11.0	-0.232204	0.0	0.0	0.0	2006.0	0.0	...	0	0	0	
4	9.0	0.373500	1.235571	30.0	-0.232204	0.0	0.0	1.0	2019.0	0.0	...	0	0	1	

5 rows × 121 columns

In [55]:
```
#Train-Test-Split (70-30 Split)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=200)
```

Models and Model Evaluation

Logistic Regression

In [56]:
```
# create data frame to collect test accuracy score values for different model types
acc_table = pd.DataFrame({"Accuracy": [0,0,0,0]},
                         index=["Logistic Regression","SVM","KNN","Random Forest"])
acc_table
```

Out[56]:

	Accuracy
Logistic Regression	0
SVM	0
KNN	0
Random Forest	0

In [57]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression(random_state=0, max_iter=20000, solver='lbfgs').fit(X_train, y_train)
y_pred = model.predict(X_test)

print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)

print('In-sample accuracy score:', accuracy_score(y_train, model.predict(X_train)))
print('Out-of-sample accuracy score:', accuracy_score(y_test, y_pred))
```

```
Coefficients: [[-1.57475199e-01 -1.29643550e-01 -1.07885672e-01 -8.77313007e-02
   4.95489117e-02  6.43557816e-03 -6.10618528e-02 -1.35891647e-02
   6.88464938e-04  3.10282112e-04  8.64310600e-02 -6.33557361e-02
  -1.94278801e-01 -1.50235218e+00 -1.36438230e+00 -9.25186635e-01
  -1.64891961e-01 -9.43750038e-01 -1.07662999e+00 -1.78634194e+00
  -1.06632374e+00  7.41295631e-01  5.87647075e-01  1.37731785e-01
   5.88830820e-01 -4.26447881e-02  5.45288661e-01 -3.68514849e-03
   6.16926398e-01 -4.86197862e-01 -2.94380694e-01  4.69261432e-01
  -5.71228721e-02 -8.66817355e-01 -4.08093944e-01 -4.29492941e-01
  -8.46293068e-01 -3.00773513e-01 -1.00983404e-02  2.34435221e-01
   2.48200165e-01  5.88830820e-01  6.27291644e-01 -3.68514849e-03
   1.02520139e+00 -1.23411855e-01  1.42512745e-01 -3.86372045e-01
  -5.83995302e-01 -6.56841927e-01 -1.98033703e-01 -4.17478183e-01
  -3.61640634e-01 -1.00983404e-02  2.34435221e-01 -6.96337185e-01
   1.84490232e-01  0.00000000e+00 -1.67913196e-01 -7.74639165e-01
  -8.05775458e-02  7.57612414e-01 -1.42140186e-01 -4.51898189e-01
   1.37399977e-01  1.13144528e+00 -5.72873252e-01 -3.06976975e-01
  -3.31773689e-01 -2.92967467e-01 -8.44986296e-01 -6.19522836e-01
  -2.69242691e-01 -5.76999889e-02 -7.11593226e-01 -2.82238903e-01
  -3.86839976e-01 -6.26029129e-01  3.27811784e-01  8.39320665e-01
  -7.03638872e-03  5.52041285e-01  1.17664852e-01  1.27469896e-01
   1.86232373e-01 -1.07141656e-01  0.00000000e+00 -2.38912653e-01
  -4.14156194e-01  0.00000000e+00 -5.90639080e-02  4.03132015e-01
  -5.02464655e-02 -2.05918908e-01  7.21423816e-01  0.00000000e+00
  -1.60033067e-01 -2.43361147e-01  0.00000000e+00 -2.80977854e-01
  -7.51491171e-01  4.81782791e-01  2.08734595e-01 -5.92668988e-01
  -6.13739764e-02  7.24338528e-02  2.82804710e-02  9.24409960e-02
  -1.96096730e-01 -2.74698630e-01  8.04674147e-02  1.24397883e+00
  -4.36275029e-01  8.73694262e-01  1.18843553e+00  3.56535198e-02
   2.56500548e-01  2.44587994e-01  9.35893185e-01  2.72004437e-01
  -1.04158269e-01]]
Intercept: [-0.03501624]
In-sample accuracy score: 0.8021680216802168
Out-of-sample accuracy score: 0.7255520504731862
```

In [58]:
```
y_prob = model.predict_proba(X_test)
print('Predicted probabilities:', y_prob)
```

```
Predicted probabilities: [[6.39490874e-01 3.60509126e-01]
 [5.74474220e-01 4.25525780e-01]
 [9.42510045e-01 5.74899547e-02]
 [4.48613261e-01 5.51386739e-01]
 [7.83864212e-01 2.16135788e-01]
 [5.56702725e-01 4.43297275e-01]
 [3.76641302e-01 6.23358698e-01]
 [4.76749838e-01 5.23250162e-01]
 [4.14305212e-01 5.85694788e-01]
 [7.71032653e-01 2.28967347e-01]
 [4.55566733e-01 5.44433267e-01]
 [4.95231308e-01 5.04768692e-01]
 [6.19069694e-01 3.80930306e-01]
 [7.27557670e-01 2.72442330e-01]
 [5.90542151e-01 4.09457849e-01]
 [6.15600731e-01 3.84399269e-01]
 [9.17066863e-01 8.29331366e-02]
 [3.15432123e-01 6.84567877e-01]
 [2.63360521e-01 7.36639479e-01]
```

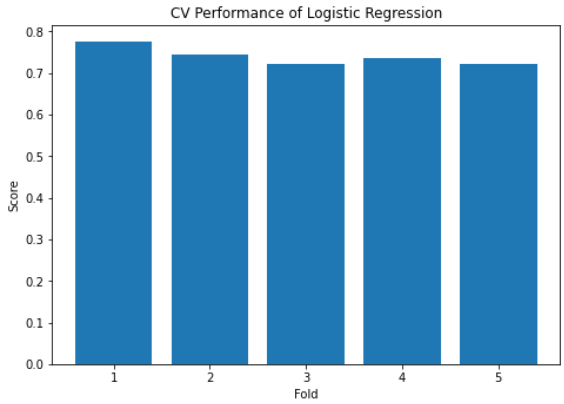
In [59]:
```
model.classes_
#probability of failure = 0
#probability of success = 1
```

Out[59]:
```
array([0, 1])
```

Model Evaluation

In [60]:
```
#Cross Validation
K=5
scores = cross_val_score(model,X_train,y_train,cv=K,scoring='accuracy')
print('Score in each fold:', scores)
print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(), scores.std()))

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(np.arange(1,K+1),scores)
plt.xlabel("Fold")
plt.ylabel("Score")
plt.title('CV Performance of Logistic Regression')
plt.show()
```

```
Score in each fold: [0.77702703 0.74324324 0.72297297 0.73469388 0.72108844]
0.74 accuracy with a standard deviation of 0.02
```



In [61]:
```
#hyperparameter tuning
from sklearn.model_selection import GridSearchCV

hyperparameters = {
    'C': [0.01, 0.1, 1.0, 10.0],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear', 'saga'],
    'max_iter': [100, 500, 1000]
}

grid = GridSearchCV(model, hyperparameters, cv=5, scoring='accuracy')

grid.fit(X_test, y_test)

print("Best hyperparameters:", grid.best_params_)
print("Accuracy score:", grid.best_score_)
```

```
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(

Best hyperparameters: {'C': 1.0, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'}
Accuracy score: 0.7100198412698413

/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
```

In [62]:
```
acc_table.loc['Logistic Regression','Accuracy'] = grid.best_score_
#Accuracy score for Logistic Regression changed
```

In [63]:
```
acc_table
```

Out[63]:

	Accuracy
Logistic Regression	0.71002
SVM	0.00000
KNN	0.00000
Random Forest	0.00000

Interpreting LG with ELI5

In [64]:
```
eli5.show_weights(grid, feature_names=np.array(X_test.columns), show_feature_values=True)
#global interpretation - Look at a model's parameters and figure out at a global level how the model works
```

Out[64]:

Error: estimator GridSearchCV(cv=5, estimator=LogisticRegression(max_iter=20000, random_state=0), param_grid={'C': [0.01, 0.1, 1.0, 10.0], 'max_iter': [100, 500, 1000], 'penalty': ['l1', 'l2'], 'solver': ['liblinear', 'saga']}, scoring='accuracy') is not supported

In [65]:
```
eli5.show_prediction(grid, X_test.iloc[1], feature_names=np.array(X_test.columns), show_feature_values=True)
#Local Interpretation: Look at a single prediction and identify features leading to that prediction
```

Out[65]:

Error: estimator GridSearchCV(cv=5, estimator=LogisticRegression(max_iter=20000, random_state=0), param_grid={'C': [0.01, 0.1, 1.0, 10.0], 'max_iter': [100, 500, 1000], 'penalty': ['l1', 'l2'], 'solver': ['liblinear', 'saga']}, scoring='accuracy') is not supported

In [66]:
```
eli5.show_weights(model, feature_names=np.array(X_test.columns), show_feature_values=True)
```

Out[66]:

y=1 top features

Weight[?]	Feature
+1.244	Industry_biotech
+1.188	Industry_hardware
+1.131	Estimated Revenue Range_N/A
+1.025	Last Equity Funding Type_Pre-Seed
+0.936	Industry_semiconductor
+0.874	Industry_games_video
+0.839	State_CT
+0.758	Estimated Revenue Range_1Mto10M
	… 37 more positive …
	… 60 more negative …
-0.751	State_NY
-0.775	Estimated Revenue Range_10Mto50M
-0.845	Founded Month_6.0
-0.846	Last Funding Type_Series E
-0.867	Last Funding Type_Series B
-0.925	Number of Employees_101-250
-0.944	Number of Employees_251-500
-1.066	Number of Employees_51-100
-1.077	Number of Employees_5001-10000
-1.364	Number of Employees_1001-5000
-1.502	Number of Employees_10001+
-1.786	Number of Employees_501-1000

In [67]:
```
eli5.show_prediction(model, X_test.iloc[1], feature_names=np.array(X_test.columns), show_feature_values=True)
```

Out[67]:

y=0 (probability **0.574**, score **-0.300**) top features

Contribution[?]	Feature	Value
+1.107	Unemployment	5.700
+1.066	Number of Employees_51-100	1.000
+0.657	Last Equity Funding Type_Series C	1.000
+0.626	State_CA	1.000
+0.472	Number of Funding Rounds	3.000
+0.439	Number of Investors	5.000
+0.408	Last Funding Type_Series C	1.000
+0.076	CPI	1.200
+0.035	<BIAS>	1.000
+0.027	Trademarks Registered	2.000
+0.012	Price Currency (in USD)	-0.232
-0.004	Total Equity Funding Amount Currency (in USD)	-0.028
-0.004	Total Funding Amount Currency (in USD)	-0.037
-0.006	Total Products Active	1.000
-0.184	IPO Status_Private	1.000
-0.257	Industry_network_hosting	1.000
-0.623	Closed Year	2009.000
-1.037	Closed Month	12.000
-1.131	Estimated Revenue Range_N/A	1.000
-1.378	Founded Year	2002.000

In [68]:
```
#Elastic Net
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import ElasticNetCV

ENcv = ElasticNetCV(alphas=None, cv=10, max_iter=10000) # default l1_ratio=0.5
ENcv.fit(X_train, y_train)

print('The best alpha from ElasticNetCV:', ENcv.alpha_)
```

```
The best alpha from ElasticNetCV: 0.1872178120019673
```

In [69]:
```
ENcv.score(X_test, y_test)
```

Out[69]:
```
0.12990274275645697
```

In [70]:
```
ENcv.predict(X_test)
```

Out[70]:
```
array([ 0.62260426,  0.52115437,  0.26858073,  0.36563675,  0.35270571,
        0.26279737,  0.53509028,  0.28293139,  0.23698109,  0.37677858,
        0.59668234,  0.36947635,  0.18817935,  0.32868839,  0.29816219,
        0.30184659,  0.12807359,  0.45542806,  0.60975245,  0.10403592,
        0.09836598,  0.3917155 ,  0.29467779,  0.3104964 ,  0.71863222,
        0.35388707,  0.28721498,  0.44550701,  0.63363069,  0.33514711,
        0.29744366,  0.3433976 , -0.01213014,  0.36407526,  0.18897548,
        0.70598366,  0.66815686,  0.55889333,  0.59214617,  0.6438673 ,
        0.49995402,  0.52311185,  0.38368965,  0.37835744,  0.30258758,
        0.73089046,  0.37925323,  0.36741492,  0.6911842 ,  0.1356248 ,
        0.24416502,  0.36140598,  0.015687  ,  0.66900491,  0.38980927,
        0.40601743,  0.31554577,  0.34981785,  0.35769953,  0.35270571,
        0.35204232,  0.32103633,  0.55519627,  0.19078039,  0.26042873,
        0.57871175,  0.65606284,  0.36765366,  0.38235184,  0.63913307,
        0.40902741,  0.25021052,  0.22863763,  0.24865699,  0.60183488,
        0.29457179,  0.31538107,  0.4303747 ,  0.36873125,  0.33373266,
        0.51732936,  0.07608551,  0.22177067,  0.59078639,  0.36496867,
        0.65139131,  0.31697814,  0.2748505 ,  0.35019164,  0.29602635,
        0.17401419,  0.31297105,  0.3301833 ,  0.26024188,  0.23274313,
        0.28720299,  0.31390286,  0.33349295,  0.32727298,  0.39931202,
        0.45419625,  0.68134075,  0.33222268,  0.39040609,  0.0887723 ,
        0.32148303,  0.29337912,  0.29499107,  0.06687125,  0.42320128,
        0.15875146,  0.16249454,  0.23817855,  0.17254646,  0.62635524,
        0.18147393,  0.35383152,  0.21364921,  0.40287939,  0.29200314,
        0.25802065,  0.4479596 ,  0.35208684,  0.23150211, -0.12276018,
        0.23011921,  0.32396431,  0.53284051,  0.68146718,  0.34155285,
        0.36741492,  0.62335628,  0.33771011,  0.38677723,  0.33039999,
        0.53221938,  0.28221014,  0.27954086,  0.29150129,  0.34654666,
        0.75370191,  0.22965045,  0.19967061,  0.28966757,  0.70569016,
        0.53611806,  0.34133616,  0.05453731,  0.18062234,  0.25968773,
        0.35086606,  0.3167138 ,  0.31953577,  0.5020008 ,  0.19296519,
        0.19118702,  0.36871484,  0.50291234,  0.24592518,  0.57219184,
        0.51075353,  0.67271808,  0.22667585,  0.20528427,  0.38104243,
        0.25419208,  0.15730576,  0.57033549,  0.08473847,  0.61535533,
        0.09737445,  0.60276888,  0.32479833,  0.13303971,  0.23121193,
        0.22622915,  0.68448982, -0.01344761,  0.31736336,  0.59533434,
        0.37925323,  0.2519672 ,  0.69959769,  0.54291598,  0.18454284,
        0.17335311,  0.38550091,  0.34155285,  0.20857653,  0.34487353,
        0.23121885,  0.56255008,  0.36015212,  0.15237417,  0.10159354,
        0.35750489,  0.21779393,  0.37878447,  0.35279434,  0.28133006,
        0.41649587,  0.29552977,  0.23881948,  0.31851911,  0.4643473 ,
        0.35894746,  0.30903962,  0.25208422,  0.268388  ,  0.54727092,
        0.11670404,  0.29925301,  0.33763124,  0.48824686,  0.47730954,
        0.1599845 ,  0.30302794,  0.14573303,  0.34963425,  0.6491156 ,
        0.268282  ,  0.6041672 ,  0.41794118,  0.40835299,  0.61050285,
        0.45732326,  0.41965003,  0.59813005,  0.35771055,  0.65103123,
        0.23589219,  0.37434804,  0.25396485,  0.2137883 ,  0.3879535 ,
        0.20255657,  0.28918809,  0.07344242,  0.15659601,  0.23902513,
        0.40095111,  0.24032069,  0.43435414,  0.31121025,  0.32702063,
        0.22198736,  0.31742484,  0.38432464,  0.52158509,  0.27705079,
        0.26050633,  0.38248499,  0.33840378,  0.30061478,  0.58710361,
        0.60061881,  0.51284481,  0.56467146,  0.3954775 ,  0.47951698,
        0.34400104,  0.44310996,  0.28978364,  0.65052551,  0.24238685,
        0.41827404,  0.34416065,  0.35692545,  0.23627924,  0.64412688,
        0.67573616,  0.29746571,  0.3529224 ,  0.15890255,  0.24270704,
        0.26878321,  0.67265151,  0.45919006,  0.68602616,  0.51732936,
        0.33526575,  0.05383626,  0.05942277,  0.25735133,  0.25634723,
        0.3879535 ,  0.34587224,  0.23067067,  0.11670404,  0.60775597,
        0.26396455,  0.56818848,  0.26778016,  0.62421983,  0.37579695,
        0.3971891 ,  0.38303729,  0.40965138,  0.36516799,  0.38301523,
        0.3897427 ,  0.23716784,  0.30363579,  0.63580753,  0.62852819,
        0.53291348,  0.64410913,  0.41990021,  0.28959507,  0.68646099,
        0.35256252,  0.53666022,  0.23168223,  0.37061413,  0.35531351,
        0.28509207,  0.58998384])
```

In [71]:
```
EN=ElasticNet()
EN.set_params(alpha=ENcv.alpha_)
EN.fit(X_train, y_train)

print('The coefficients are:')
print(pd.Series(EN.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                        -0.004242
Total Equity Funding Amount Currency (in USD)   -0.000000
Total Funding Amount Currency (in USD)          -0.000000
Number of Investors                             -0.019853
Price Currency (in USD)                         -0.000000
                                                   ...
Industry_network_hosting                        -0.000000
Industry_other                                  -0.000000
Industry_semiconductor                           0.000000
Industry_software                               -0.000000
Industry_web                                    -0.000000
Length: 121, dtype: float64
```

In [72]:
```
 Regression with Elastic Net

rn.linear_model import LogisticRegressionCV
rn.metrics import classification_report

gisticRegression algorithm
egression_classifier = LogisticRegressionCV(cv=3)

t_classifier = LogisticRegressionCV(cv=3, penalty='elasticnet', l1_ratios=[0.1, 0.5, 0.9], solver='saga')

e models
egression_classifier.fit(X_train, y_train)
t_classifier.fit(X_train, y_train)

 models
istic Regression: {} || Elasticnet: {}".format(logistic_regression_classifier.score(X_test, y_test), elastic_net_classifier.score(X_t

t some more metrics
istic Regression")
sification_report(y_test, logistic_regression_classifier.predict(X_test)))
stic Net")
sification_report(y_test, elastic_net_classifier.predict(X_test)))
```

```
  warnings.warn(

/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(

Logistic Regression: 0.6782334384858044 || Elasticnet: 0.6624605678233438
Logistic Regression
```

SVM

In [73]:
```
from sklearn.svm import SVC
import numpy as np

model = SVC(kernel='linear', C=1, random_state=0)
model.fit(X_train, y_train)

print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)

from sklearn.metrics import accuracy_score
print('In-sample accuracy score:',accuracy_score(y_train, model.predict(X_train)))
print('Out-of-sample accuracy score:',accuracy_score(y_test, model.predict(X_test)))
```

```
Coefficients: [[-6.75434859e-01 -7.68156543e-01  1.80339654e-01 -1.69203981e-01
   1.05965674e-01 -1.94083125e-03 -9.38021920e-02 -1.90073429e-03
   3.24065973e-03  4.63391551e-04  2.57366826e-01 -4.71280672e-01
  -3.31183381e-01 -1.47728141e+00 -7.71711710e-01 -2.47368928e+00
  -3.72760833e-01 -1.13529218e+00 -3.00000000e+00 -4.01531232e+00
  -2.05407649e+00  2.00000000e+00  1.78968004e+00  5.58662532e-01
   1.00000000e+00  1.52766702e-01  1.15634737e+00  0.00000000e+00
   1.22536075e+00 -1.09081471e-01 -1.00000000e+00  6.26082895e-01
  -6.62421484e-01 -2.94125207e+00 -3.52156579e-01 -7.75030360e-01
  -1.21274254e+00 -4.46012015e-01  0.00000000e+00  0.00000000e+00
   1.20426118e+00  1.00000000e+00  1.15634737e+00  0.00000000e+00
   1.37812745e+00 -1.09081471e-01  6.26082895e-01 -6.62421484e-01
   5.87479319e-02 -1.56247654e+00 -3.42504717e-01 -4.95613346e-01
   1.69543025e-01  0.00000000e+00  0.00000000e+00 -2.00228617e+00
   7.90478794e-14  0.00000000e+00 -7.19075027e-01 -3.91499662e-01
   0.00000000e+00  1.06779857e+00  0.00000000e+00 -2.00000000e+00
  -3.16722625e-01  2.35949875e+00 -7.19075027e-01 -7.64013602e-01
  -1.17171535e+00 -1.00000000e+00 -8.63718451e-01 -1.17665459e+00
  -5.65337840e-01  0.00000000e+00 -4.58370947e-01  1.20491078e-01
   1.39459267e-01 -1.73758515e+00  8.90918529e-01  1.40895764e+00
  -9.90063391e-01  1.00000000e+00 -1.53123195e-02  4.58725674e-02
   5.30058436e-01  0.00000000e+00  0.00000000e+00  0.00000000e+00
  -8.73503777e-01  0.00000000e+00  0.00000000e+00  1.00000000e+00
   1.00000000e+00 -4.69996245e-01  2.43252564e+00  0.00000000e+00
   0.00000000e+00 -2.06547348e-01  0.00000000e+00 -1.00000000e+00
  -1.10031896e+00  0.00000000e+00  0.00000000e+00 -2.92945290e-01
   0.00000000e+00  0.00000000e+00  5.55311377e-01 -6.60880678e-01
  -1.00000000e+00 -5.16491031e-01  0.00000000e+00  3.16271088e+00
  -5.15803104e-01  2.07907870e+00  3.08368293e+00  7.33321929e-01
  -6.44340159e-01  6.23862733e-01  2.72140632e+00  1.15087218e+00
  -3.34486622e-01]]
Intercept: [-2.51943406]
In-sample accuracy score: 0.7899728997289973
Out-of-sample accuracy score: 0.7160883280757098
```

In [74]:
```
svm_acc = accuracy_score(y_test, model.predict(X_test))
```

In [75]:
```
acc_table.loc['SVM','Accuracy'] = svm_acc
```

In [76]:
```
acc_table
```

Out[76]:

	Accuracy
Logistic Regression	0.710020
SVM	0.716088
KNN	0.000000
Random Forest	0.000000

Interpreting SVM with ELI5

In [77]:
```
eli5.show_weights(model,feature_names=np.array(X_test.columns))
```

Out[77]: **y=1** top features

Weight[?]	Feature
+3.163	Industry_biotech
+3.084	Industry_hardware
+2.721	Industry_semiconductor
+2.433	State_NC
+2.359	Estimated Revenue Range_N/A
+2.079	Industry_games_video
+2.000	Number of Employees_N/A
+1.790	Last Funding Type_Convertible Note
+1.409	State_CT
	… 32 more positive …
	… 47 more negative …
-1.477	Number of Employees_10001+
-1.562	Last Equity Funding Type_Series C
-1.738	State_CA
-2.000	Estimated Revenue Range_50Mto100M
-2.002	Last Equity Funding Type_Venture - Series Unknown
-2.054	Number of Employees_51-100
-2.474	Number of Employees_101-250
-2.519	<BIAS>
-2.941	Last Funding Type_Series B
-3.000	Number of Employees_5001-10000
-4.015	Number of Employees_501-1000

In [78]:
```
eli5.show_prediction(model, X_test.iloc[1], feature_names=np.array(X_test.columns), show_feature_values=True)
```

Out[78]: **y=0** (score **-1.344**) top features

Contribution[?]	Feature	Value
+2.519	<BIAS>	1.000
+2.054	Number of Employees_51-100	1.000
+2.026	Number of Funding Rounds	3.000
+1.888	Unemployment	5.700
+1.738	State_CA	1.000
+1.562	Last Equity Funding Type_Series C	1.000
+0.846	Number of Investors	5.000
+0.644	Industry_network_hosting	1.000
+0.566	CPI	1.200
+0.352	Last Funding Type_Series C	1.000
+0.025	Price Currency (in USD)	-0.232
+0.007	Total Funding Amount Currency (in USD)	-0.037
+0.004	Trademarks Registered	2.000
+0.002	Total Products Active	1.000
-0.000	IPO Status_Private	1.000
-0.022	Total Equity Funding Amount Currency (in USD)	-0.028
-0.931	Closed Year	2009.000
-2.359	Estimated Revenue Range_N/A	1.000
-3.088	Closed Month	12.000
-6.488	Founded Year	2002.000

In [79]:
```
y_test[1]
```

Out[79]: 1

In [80]:
```
y_pred[1]
```

Out[80]: 0

In [81]:
```
eli5.show_prediction(model, X_test.iloc[36], feature_names=np.array(X_test.columns), show_feature_values=True)
```

Out[81]: **y=1** (score **3.826**) top features

Contribution[?]	Feature	Value
+6.507	Founded Year	2008.000
+3.088	Closed Month	12.000
+2.359	Estimated Revenue Range_N/A	1.000
+1.204	Last Funding Type_Venture - Series Unknown	1.000
+0.931	Closed Year	2010.000
+0.624	Industry_other	1.000
+0.099	Total Equity Funding Amount Currency (in USD)	-0.129
+0.000	IPO Status_Private	1.000
-0.008	Total Products Active	4.000
-0.021	Total Funding Amount Currency (in USD)	-0.115
-0.025	Price Currency (in USD)	-0.232
-0.169	Number of Investors	1.000
-0.516	State_WA	1.000
-1.351	Number of Funding Rounds	2.000
-2.002	Last Equity Funding Type_Venture - Series Unknown	1.000
-2.020	Unemployment	6.100
-2.356	CPI	5.000
-2.519	<BIAS>	1.000

In [82]:
```
y_test[36]
```

Out[82]: 1

In [83]:
```
y_pred[36]
```

Out[83]: 1

KNN

In [84]:
```
from sklearn.neighbors import KNeighborsClassifier

acc_train = [-1]*30
acc_test = [-1]*30
```

In [85]:
```
for K in range(30):
    model = KNeighborsClassifier(n_neighbors = K+1)
    model.fit(X_train,y_train)  #fit the model
    acc_train[K] = accuracy_score(y_train, model.predict(X_train))
    acc_test[K] = accuracy_score(y_test, model.predict(X_test))
```

In [86]:
```
# optimal k (max accuracy value)
np.argmax(acc_test)+1
```

Out[86]: 9

In [87]:
```
# optimal accuracy
max(acc_test)
```

Out[87]: 0.7192429022082019

In [88]:
```
acc_table.loc['KNN','Accuracy'] = max(acc_test)
```

Random Forest

Type *Markdown* and LaTeX: α^2

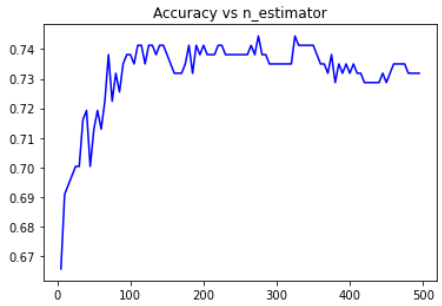
In [89]:
```
from sklearn.ensemble import RandomForestClassifier
from pprint import pprint
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

In [90]:
```
class_RF = RandomForestClassifier(n_estimators=10, random_state=0).fit(scale(X_train), y_train)
```

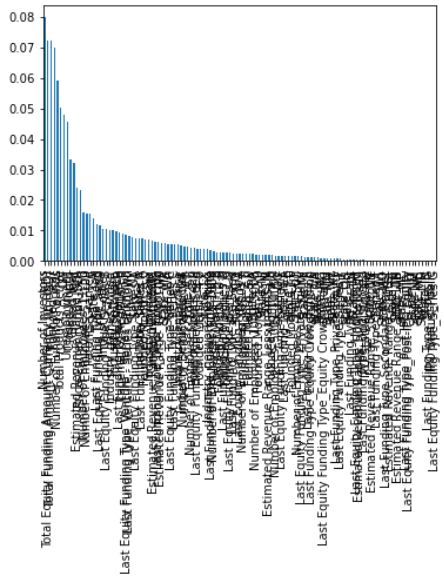
In [91]:
```
y_pred=class_RF.predict(scale(X_test.values))
acc_rf = accuracy_score(y_test, y_pred)
print(f"Accuracy for Random Forest is {acc_rf}")
```

```
Accuracy for Random Forest is 0.6908517350157729
```

In [92]:
```
acc_list = []
n_list = np.arange(5,500,5)
for n in n_list:
    class_RF = RandomForestClassifier(n_estimators=n,random_state=0).fit(scale(X_train), y_train)
    y_pred=class_RF.predict(scale(X_test.values))
    acc = accuracy_score(y_test, y_pred)
    acc_list.append(acc)

plt.plot(n_list, acc_list,  color = 'blue', markerfacecolor = 'black',label = 'Accuracy')
plt.title('Accuracy vs n_estimator')
plt.show()
```



In [93]:
```
# Create a series containing feature importances from the model and feature names from the training data
feature_importances = pd.Series(class_RF.feature_importances_, index=X_train.columns).sort_values(ascending=False)

# Plot a simple bar chart
feature_importances.plot.bar();
```



In [94]:
```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['sqrt', 'log2',None]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 100, num = 10)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
 'max_features': ['sqrt', 'log2', None],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [100, 311, 522, 733, 944, 1155, 1366, 1577, 1788, 2000]}
```

In [95]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, r
# Fit the random search model
rf_random.fit(scale(X_train), y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
[CV] END bootstrap=True, max_depth=100, max_features=log2, min_samples_leaf=1, min_samples_split=10, n_estimators=733; total time=
1.1s
[CV] END bootstrap=True, max_depth=80, max_features=None, min_samples_leaf=1, min_samples_split=5, n_estimators=1577; total time=
6.5s
[CV] END bootstrap=True, max_depth=80, max_features=None, min_samples_leaf=2, min_samples_split=10, n_estimators=1577; total time=
6.5s
[CV] END bootstrap=True, max_depth=70, max_features=None, min_samples_leaf=4, min_samples_split=2, n_estimators=2000; total time=
6.9s
[CV] END bootstrap=False, max_depth=None, max_features=None, min_samples_leaf=4, min_samples_split=10, n_estimators=1155; total ti
me=   5.1s
[CV] END bootstrap=True, max_depth=30, max_features=log2, min_samples_leaf=2, min_samples_split=10, n_estimators=311; total time=
0.5s
[CV] END bootstrap=True, max_depth=30, max_features=log2, min_samples_leaf=2, min_samples_split=10, n_estimators=311; total time=
0.4s
[CV] END bootstrap=False, max_depth=70, max_features=None, min_samples_leaf=1, min_samples_split=10, n_estimators=100; total time=
0.6s
[CV] END bootstrap=True, max_depth=10, max_features=None, min_samples_leaf=2, min_samples_split=10, n_estimators=733; total time=
2.7s
```

In [96]:
```
rf_random.best_params_
```

Out[96]:
```
{'n_estimators': 311,
 'min_samples_split': 10,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 90,
 'bootstrap': True}
```

In [97]:
```
best_rd = rf_random.best_estimator_
```

In [98]:
```
y_pred=best_rd.predict(scale(X_test.values))
acc_best_rf = accuracy_score(y_test, y_pred)
print(f"Accuracy is {acc_best_rf}")
```

```
Accuracy is 0.7444794952681388
```

In [99]:
```
acc_table.loc['Random Forest','Accuracy'] = acc_best_rf
```

In [100]:
```
acc_table
```

Out[100]:

	Accuracy
Logistic Regression	0.710020
SVM	0.716088
KNN	0.719243
Random Forest	0.744479

In [101]:
```
y_prob = best_rd.predict_proba(X_test)
print('Predicted probabilities:', y_prob)
```

```
Predicted probabilities: [[0.55397626 0.44602374]
 [0.5419288  0.4580712 ]
 [0.47094656 0.52905344]
 [0.37748919 0.62251081]
 [0.41550795 0.58449205]
 [0.48548746 0.51451254]
 [0.58093352 0.41906648]
 [0.36871494 0.63128506]
 [0.44432955 0.55567045]
 [0.4232948  0.5767052 ]
 [0.47897896 0.52102104]
 [0.4998876  0.5001124 ]
 [0.48004681 0.51995319]
 [0.43527278 0.56472722]
 [0.39819282 0.60180718]
 [0.38731676 0.61268324]
 [0.43180103 0.56819897]
 [0.4755144  0.5244856 ]
 [0.43852062 0.56147938]
```

In [102]:
```
X_test
```

Out[102]:

	Number of Funding Rounds	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	Total Products Active	Patents Granted	Trademarks Registered	Founded Year	Closed Year	...	Industry_biotech	Industry_enterprise	Industry_games_video	Indus
500	2.0	1.680108	3.058974	3.0	2.214322	0.0	0.0	0.0	2007.0	2013.0	...	0	0	0	
666	3.0	-0.028121	-0.037109	5.0	-0.232204	1.0	0.0	2.0	2002.0	2009.0	...	0	0	0	
50	10.0	-0.142985	-0.122031	6.0	-0.115908	23.0	0.0	0.0	2009.0	0.0	...	0	1	0	
227	2.0	-0.142717	-0.136147	3.0	-0.232204	2.0	0.0	0.0	2007.0	0.0	...	0	0	0	
479	3.0	-0.154364	-0.146212	3.0	-0.232204	0.0	0.0	0.0	2006.0	0.0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
413	4.0	-0.022767	-0.032481	6.0	-0.102986	0.0	1.0	0.0	2001.0	0.0	...	0	0	0	
1016	2.0	-0.150214	-0.142626	3.0	-0.232204	5.0	17.0	2.0	2010.0	0.0	...	0	0	0	
143	3.0	-0.037225	-0.044976	2.0	1.275339	14.0	0.0	0.0	2006.0	0.0	...	0	0	0	
572	2.0	-0.127188	-0.122726	4.0	-0.124522	6.0	0.0	0.0	2002.0	0.0	...	0	0	0	
627	6.0	-0.150991	-0.142371	3.0	-0.232204	0.0	3.0	0.0	2006.0	2012.0	...	0	0	0	

317 rows × 121 columns

Interpreting RF with ELI5

In [103]:
```
eli5.show_weights(best_rd, feature_names=np.array(X_test.columns))
#eli5.show_weights(best_rd, feature_names=np.array(X_test.columns), top=121) shows all variables
```

Out[103]:

Weight	Feature
0.0927 ± 0.1173	Number of Investors
0.0762 ± 0.0812	Founded Year
0.0704 ± 0.1026	Number of Funding Rounds
0.0694 ± 0.0744	Total Funding Amount Currency (in USD)
0.0664 ± 0.0676	Total Equity Funding Amount Currency (in USD)
0.0419 ± 0.0761	Closed Year
0.0393 ± 0.0473	Total Products Active
0.0392 ± 0.0514	CPI
0.0381 ± 0.0438	Unemployment
0.0359 ± 0.0709	Closed Month
0.0305 ± 0.0589	Estimated Revenue Range_N/A
0.0264 ± 0.0393	Price Currency (in USD)
0.0171 ± 0.0368	Number of Employees_51-100
0.0158 ± 0.0537	Last Equity Funding Type_Seed
0.0136 ± 0.0249	Patents Granted
0.0131 ± 0.0416	Last Funding Type_Seed
0.0110 ± 0.0244	Industry_games_video
0.0107 ± 0.0262	Last Equity Funding Type_Series B
0.0100 ± 0.0227	Number of Employees_11-50
0.0098 ± 0.0245	Last Funding Type_Series B
	… 101 more …

In [104]:
```
eli5.show_prediction(best_rd, X_test.iloc[36], feature_names=np.array(X_test.columns), show_feature_values=True)
#this shows individual prediction for index row 36 (i just chose a random row)
```

```
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but RandomForestClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but Decis
```

In [105]:
```
y_test[36]
```

Out[105]: 1

In [106]:
```
y_pred[36]
```

Out[106]: 1

In [107]:
```
eli5.show_prediction(best_rd, X_test.iloc[1], feature_names=np.array(X_test.columns), show_feature_values=True)
```

```
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but RandomForestClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but DecisionTreeClassifier was fitted without feature names
  warnings.warn(
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but Decis
```

In [108]:
```
y_test[1]
```

Out[108]: 1

In [109]:
```
y_pred[1]
```

Out[109]: 0

Model Comparison - Classifiers

In [110]:
```
acc_table
#Random forest is the most accurate model
```

Out[110]:

	Accuracy
Logistic Regression	0.710020
SVM	0.716088
KNN	0.719243
Random Forest	0.744479

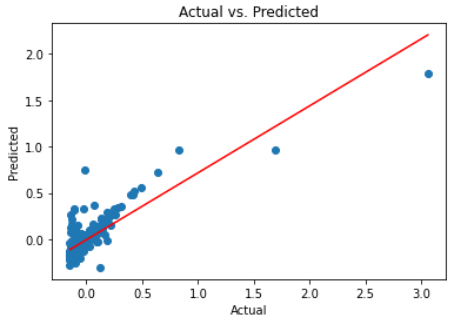
Note: Due to random state changes, model accuracy changes.

Linear Regression - For Total Funding Amount

In [111]:
```
# Define X and y
X = merged_df_test.drop(columns=['Total Funding Amount Currency (in USD)'], axis=1)
y = merged_df_test['Total Funding Amount Currency (in USD)']

for col in X.select_dtypes(include='object').columns:
    X[col] = X[col].astype('category')
X = pd.get_dummies(data=X, drop_first=True)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=200)


# Create LinearRegression object and fit to training data
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

# Predict y values for test data
y_pred = linear_regressor.predict(X_test)

# Display coefficients
coefficients = pd.Series(linear_regressor.coef_.flatten(), index=X_train.columns)
print("Coefficients:")
print(coefficients)

# Calculate training and test MSE
mse_train = mean_squared_error(y_train, linear_regressor.predict(X_train))
mse_test = mean_squared_error(y_test, y_pred)

print("Training MSE:", mse_train)
print("Test MSE:", mse_test)
print("MSE:", round(mean_squared_error(y_test, y_pred), 3))
```

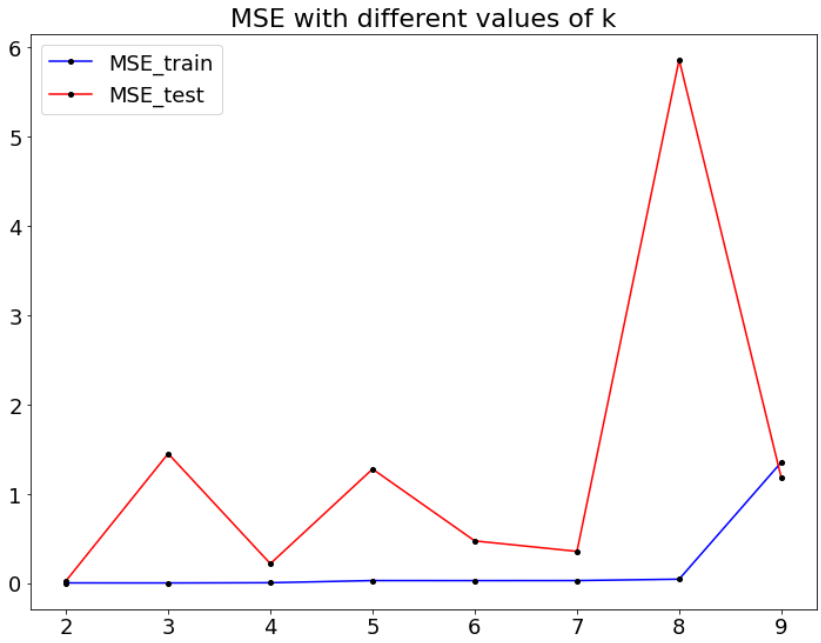
```
Coefficients:
Number of Funding Rounds                          0.000569
Total Equity Funding Amount Currency (in USD)     1.001847
Number of Investors                               0.000824
Price Currency (in USD)                          -0.010034
Total Products Active                            -0.000189
                                                    ...
Industry_other                                   -0.003161
Industry_semiconductor                            0.006109
Industry_software                                 0.011877
Industry_web                                      0.000929
Acquisition_Not Acquired                          0.002508
Length: 121, dtype: float64
Training MSE: 0.0055476771585391
Test MSE: 0.015011896011066611
MSE: 0.015
```

In [112]:
```
plt.scatter(y_test, y_pred)
z = np.polyfit(y_test, y_pred, 1)
p = np.poly1d(z)
plt.plot(y_test, p(y_test), color='red')
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```



In [113]:
```
k_value = [k for k in range(2, 10)]
MSE_train = []
MSE_test = []

x_train_extend = X_train
x_test_extend = X_test

for k in k_value:

    # Add x^k into the training and test data
    x_train_extend = np.concatenate((x_train_extend, X_train**k), axis=1)
    x_test_extend = np.concatenate((x_test_extend, X_test**k), axis=1)

    # Train the model using the training sets
    linear_regressor.fit(x_train_extend,y_train)

    # Record training MSE
    MSE_train.append(mean_squared_error(y_train, linear_regressor.predict(x_train_extend)))

    # Record test MSE
    MSE_test.append(mean_squared_error(y_test, linear_regressor.predict(x_test_extend)))
```

In [114]:
```
import matplotlib.pyplot as plt
# plot the train and test MSE values for various orders of K
# select model with minimum test MSE
fig, ax1 = plt.subplots(figsize=(12, 9))
plt.plot(k_value, MSE_train,  color = 'blue', marker = '.', markersize = 8, markeredgecolor = 'black', markerfacecolor = 'black',labe
plt.plot(k_value, MSE_test,  color = 'red',marker = '.', markersize = 8, markeredgecolor = 'black', markerfacecolor = 'black', label
plt.legend(fontsize=18)
plt.title('MSE with different values of k', fontsize=22)
plt.tick_params(labelsize=18)
plt.show()
```

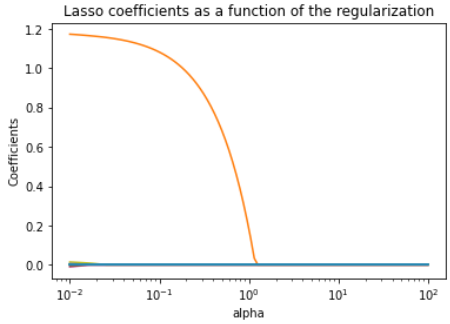


In [115]:
```
MSE_test
```

Out[115]:
```
[0.027150970412346334,
 1.4522453008159744,
 0.21953106666125322,
 1.279591694986612,
 0.47477954573744596,
 0.35763870114086044,
 5.862068077439893,
 1.1793496464394213]
```

Model Evaluation

In [116]:
```
alphas = 10**np.linspace(-2,2,100)

lasso = Lasso()
coefs = []

for a in alphas:
    lasso.set_params(alpha=a, max_iter=10000) # increase iterations for optimization of coefficients
    lasso.fit(scale(X_train), y_train)
    coefs.append(lasso.coef_)

ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
ax.set_xlim(ax.get_xlim())
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('Coefficients')
# plt.legend(list(X_train.columns), loc='best')

plt.title('Lasso coefficients as a function of the regularization');
```



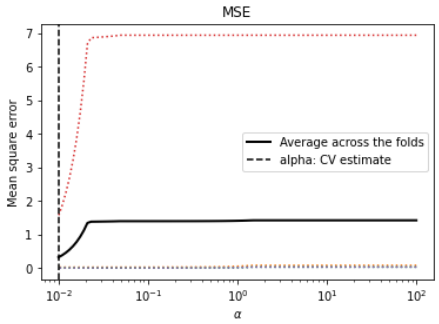
In [117]:
```
lassocv = LassoCV(alphas=alphas, max_iter=10000)
lassocv.fit(scale(X_train), y_train)

print('The best alpha from LassoCV:', lassocv.alpha_)
```

```
The best alpha from LassoCV: 0.01
```

In [118]:
```
lasso.set_params(alpha=lassocv.alpha_)
lasso.fit(scale(X_train), y_train)
print('The coefficients are:')
print(pd.Series(lasso.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                         -0.000000
Total Equity Funding Amount Currency (in USD)     1.173946
Number of Investors                              -0.000000
Price Currency (in USD)                          -0.000000
Total Products Active                            -0.000000
                                                     ...
Industry_other                                   -0.000000
Industry_semiconductor                            0.000000
Industry_software                                 0.000000
Industry_web                                     -0.000000
Acquisition_Not Acquired                          0.000000
Length: 121, dtype: float64
```

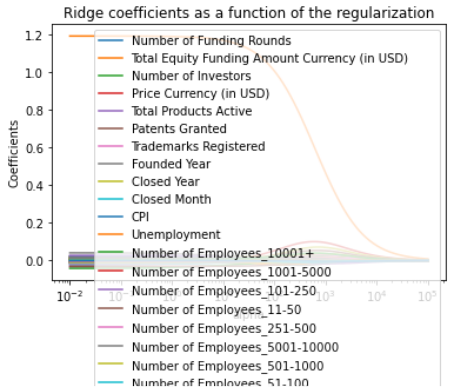
In [119]:
```
plt.semilogx(lassocv.alphas_, lassocv.mse_path_, linestyle=":")
plt.plot(
    lassocv.alphas_,
    lassocv.mse_path_.mean(axis=-1),
    color="black",
    label="Average across the folds",
    linewidth=2,
)
plt.axvline(lassocv.alpha_, linestyle="--", color="black", label="alpha: CV estimate")


plt.xlabel(r"$\alpha$")
plt.ylabel("Mean square error")
plt.legend()
plt.title("MSE")
plt.show()
```



In [120]:
```
lasso.set_params(alpha=lassocv.alpha_)
y_pred=lasso.predict(scale(X_test.values))
mse = round(mean_squared_error(y_test, y_pred),3)
print(f"MSE is {mse}")
```

```
MSE is 0.923
```

In [121]:
```
# ridge regression model
alphas = 10**np.linspace(-2,5,100)

ridge = Ridge()
coefs = []

for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(scale(X_train), y_train)
    coefs.append(ridge.coef_)

ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
ax.set_xlim(ax.get_xlim())
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('Coefficients')
plt.legend(list(X.columns), loc='best')

plt.title('Ridge coefficients as a function of the regularization');
```



In [122]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler().fit(X_train)
```

In [123]:
```
# set a large alpha to get smaller coefficients
ridge = Ridge(alpha=1000)
ridge.fit(scaler.transform(X_train), y_train)

print('The coefficients are:')
print(pd.Series(ridge.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                          0.003063
Total Equity Funding Amount Currency (in USD)     0.465521
Number of Investors                               0.022936
Price Currency (in USD)                           0.095144
Total Products Active                            -0.007740
                                                    ...
Industry_other                                   -0.005006
Industry_semiconductor                           -0.004573
Industry_software                                -0.000504
Industry_web                                     -0.004252
Acquisition_Not Acquired                         -0.004078
Length: 121, dtype: float64
```

In [124]:
```
ridgecv = RidgeCV(alphas=alphas, scoring='neg_mean_squared_error')
ridgecv.fit(scale(X_train), y_train)

print('The best alpha from RidgeCV:', ridgecv.alpha_)
```

```
The best alpha from RidgeCV: 0.01
```

In [125]:
```
ridge.set_params(alpha=ridgecv.alpha_)
ridge.fit(scale(X_train), y_train)
print('The coefficients are:')
print(pd.Series(ridge.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                          0.001114
Total Equity Funding Amount Currency (in USD)     1.192815
Number of Investors                               0.003113
Price Currency (in USD)                          -0.011214
Total Products Active                            -0.002068
                                                    ...
Industry_other                                   -0.001355
Industry_semiconductor                            0.001188
Industry_software                                 0.004324
Industry_web                                      0.000341
Acquisition_Not Acquired                          0.001202
Length: 121, dtype: float64
```

In [126]:
```
y_pred=ridge.predict(scale(X_test.values))
mse = round(mean_squared_error(y_test, y_pred),3)
print(f"MSE is {mse}")
```

```
MSE is 0.955
```

In [127]:
```
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import ElasticNetCV

ENcv = ElasticNetCV(alphas=None, cv=10, max_iter=10000) # default l1_ratio=0.5
ENcv.fit(scale(X_train), y_train)

print('The best alpha from ElasticNetCV:', ENcv.alpha_)
```

```
The best alpha from ElasticNetCV: 0.0023655233980771117
```

In [128]:
```
# with the best alpha
EN=ElasticNet()
EN.set_params(alpha=ENcv.alpha_)
EN.fit(scale(X_train), y_train)
en_coefs = pd.Series(EN.coef_.flatten(), index=X_train.columns)

print('The coefficients are:')
print(en_coefs)
```

```
The coefficients are:
Number of Funding Rounds                        -0.000000
Total Equity Funding Amount Currency (in USD)    1.187422
Number of Investors                              0.000000
Price Currency (in USD)                         -0.005941
Total Products Active                           -0.001117
                                                   ...
Industry_other                                  -0.000000
Industry_semiconductor                           0.000000
Industry_software                                0.003226
Industry_web                                     0.000000
Acquisition_Not Acquired                         0.000262
Length: 121, dtype: float64
```

In [129]:
```
y_pred=EN.predict(scale(X_test.values))
#best_model = y_pred
mse = round(mean_squared_error(y_test, y_pred),3)
print(f"MSE is {mse}")
```

```
MSE is 0.948
```

In [130]:
```
EN_array = pd.DataFrame(en_coefs)
pd.set_option('display.max_rows', None)
```

In [131]:
```
EN_array
```

Out[131]:

	0
Number of Funding Rounds	-0.000000
Total Equity Funding Amount Currency (in USD)	1.187422
Number of Investors	0.000000
Price Currency (in USD)	-0.005941
Total Products Active	-0.001117
Patents Granted	0.014403
Trademarks Registered	-0.030899
Founded Year	0.005261
Closed Year	-0.000000
Closed Month	-0.000000
CPI	0.000239

In [132]:
```
len(EN_array)
```

Out[132]:
```
121
```

Linear Regression - For Acquired Companies Only

In [133]:
```
value_counts = merged_df_test['Acquisition'].value_counts()
print(merged_df_test['Acquisition'].value_counts())
```

```
Acquired        676
Not Acquired    379
Name: Acquisition, dtype: int64
```

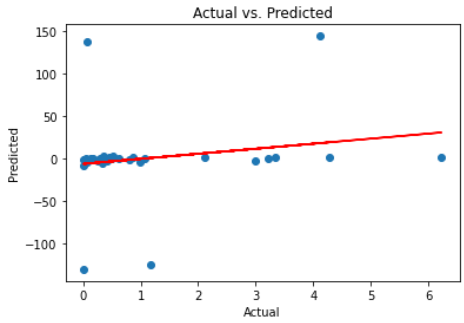
In [134]:
```
#Subsetting the data to only include companies that were acquired AND have a price that is greater than 0
df_acquired = merged_df_test[merged_df_test["Acquisition"] == "Acquired"]
df_price = df_acquired[df_acquired["Price Currency (in USD)"] > 0]
```

In [135]:
```
# Define X and y
X1 = df_price.drop(columns=['Price Currency (in USD)'])
y1 = df_price['Price Currency (in USD)']

for col in X1.select_dtypes(include='object').columns:
    X1[col] = X1[col].astype('category')
X1 = pd.get_dummies(data=X1, drop_first=True)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3, random_state=200)
```

In [136]:
```
# Create LinearRegression object and fit to training data
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

# Predict y values for test data
y_pred = linear_regressor.predict(X_test)

# Display coefficients
coefficients = pd.Series(linear_regressor.coef_.flatten(), index=X_train.columns)
print("Coefficients:")
print(coefficients)

# Calculate training and test MSE
mse_train = mean_squared_error(y_train, linear_regressor.predict(X_train))
mse_test = mean_squared_error(y_test, y_pred)

print("Training MSE:", mse_train)
print("Test MSE:", mse_test)
print("MSE:", round(mean_squared_error(y_test, y_pred), 3))
```

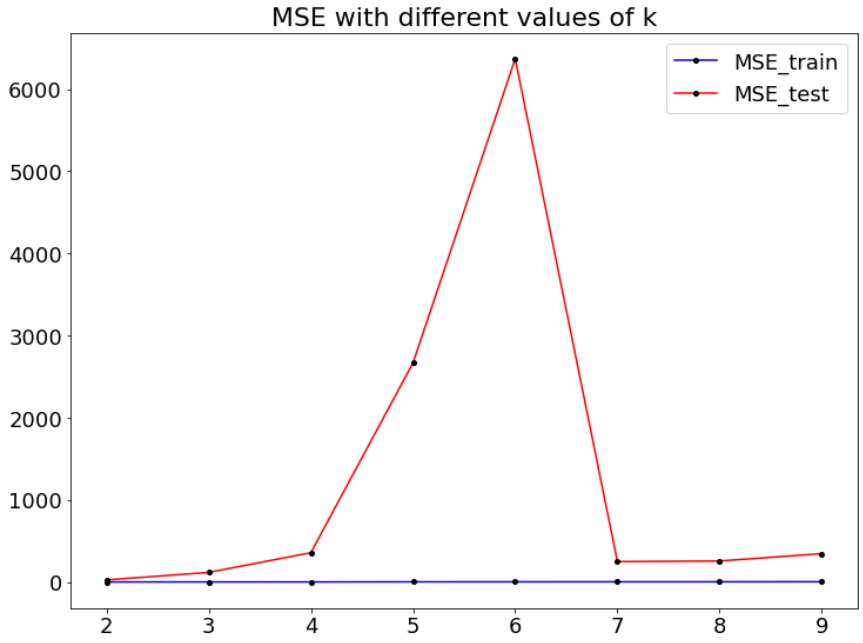
```
Coefficients:
Number of Funding Rounds                              -6.282463e-01
Total Equity Funding Amount Currency (in USD)         -2.108959e+00
Total Funding Amount Currency (in USD)                 6.483919e+00
Number of Investors                                    6.109562e-02
Total Products Active                                  8.071414e-03
Patents Granted                                       -5.712632e-02
Trademarks Registered                                  1.605778e-01
Founded Year                                           1.816336e-03
Closed Year                                            2.409577e-04
Closed Month                                          -3.774338e-01
CPI                                                   -3.123141e-01
Unemployment                                           4.750371e-01
Number of Employees_10001+                             4.544730e+00
Number of Employees_1001-5000                          8.367669e+00
Number of Employees_101-250                            2.107166e+00
Number of Employees_11-50                              1.007624e+00
Number of Employees_251-500                           -3.862498e-01
Number of Employees_5001-10000                        -5.591537e-01
Number of Employees_501-1000                           1.093001e+00
Number of Employees_51-100                             6.440037e-01
Last Funding Type_Grant                                1.801635e+00
Last Funding Type_Post-IPO Equity                     -4.966165e+00
Last Funding Type_Private Equity                      -5.301746e-01
Last Funding Type_Secondary Market                     1.452135e+00
Last Funding Type_Seed                                -3.455028e+00
Last Funding Type_Series A                            -9.278254e+00
Last Funding Type_Series B                            -5.190080e-01
Last Funding Type_Series C                            -2.660948e+00
Last Funding Type_Series D                             4.943830e-01
Last Funding Type_Series E                            -2.757177e-01
Last Funding Type_Series F                            -3.028130e-02
Last Funding Type_Series G                             4.478142e-02
Last Funding Type_Venture - Series Unknown            -1.132339e-01
Last Equity Funding Type_Private Equity               -5.301746e-01
Last Equity Funding Type_Seed                         -3.455028e+00
Last Equity Funding Type_Series A                      6.805921e+00
Last Equity Funding Type_Series B                     -5.190080e-01
Last Equity Funding Type_Series C                      1.092389e+00
Last Equity Funding Type_Series D                      4.943830e-01
Last Equity Funding Type_Series E                      1.176417e+00
Last Equity Funding Type_Series F                     -3.028130e-02
Last Equity Funding Type_Series G                      4.478141e-02
Last Equity Funding Type_Venture - Series Unknown     -1.132339e-01
IPO Status_Private                                     2.200983e+00
Estimated Revenue Range_$10B+                          1.271966e+02
Estimated Revenue Range_$10M to $50M                   2.086398e+00
Estimated Revenue Range_$1M to $10M                    2.055924e+00
Estimated Revenue Range_$500M to $1B                   5.165327e+00
Estimated Revenue Range_$50M to $100M                 -2.082445e+00
Estimated Revenue Range_Less than $1M                 -6.081150e-01
Estimated Revenue Range_N/A                            1.613245e+00
Founded Month_2.0                                      1.461787e+00
Founded Month_3.0                                      1.136805e+00
Founded Month_4.0                                      2.031997e+00
Founded Month_5.0                                     -4.133620e+00
Founded Month_6.0                                     -3.719843e-01
Founded Month_7.0                                      1.306247e+02
Founded Month_8.0                                      5.535082e-01
Founded Month_9.0                                      7.432144e-02
Founded Month_10.0                                     1.041406e+00
Founded Month_11.0                                     1.826527e+00
Founded Month_12.0                                    -6.081150e-01
State_CO                                              -2.364555e-01
State_CT                                              -4.966165e+00
State_GA                                               1.204461e+00
State_MA                                               6.097502e-01
State_MD                                               3.768229e+00
State_ME                                               7.208643e-14
State_MN                                              -5.285382e+00
State_NC                                              -1.289757e+02
State_NJ                                               2.486900e-14
State_NV                                               5.684342e-14
State_NY                                               3.143700e-01
State_OR                                               0.000000e+00
State_PA                                              -1.497842e-01
State_TX                                              -6.131415e+00
State_VA                                               6.600380e-01
State_WA                                              -2.702650e-01
Industry_biotech                                      -1.619726e+00
Industry_enterprise                                   -1.561557e+00
Industry_games_video                                  -1.292419e+02
Industry_hardware                                      2.541215e+00
Industry_mobile                                       -6.252127e-01
Industry_other                                         2.206024e-01
Industry_semiconductor                                -2.411149e+00
Industry_software                                      1.632954e-01
Industry_web                                          -1.452426e+00
dtype: float64
Training MSE: 0.01225359187283394
Test MSE: 2090.784877371829
MSE: 2090.785
```

In [137]:
```
plt.scatter(y_test, y_pred)
z = np.polyfit(y_test, y_pred, 1)
p = np.poly1d(z)
plt.plot(y_test, p(y_test), color='red')
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```



In [138]:
```
k_value = [k for k in range(2, 10)]
MSE_train = []
MSE_test = []

x_train_extend = X_train
x_test_extend = X_test

for k in k_value:

    # Add x^k into the training and test data
    x_train_extend = np.concatenate((x_train_extend, X_train**k), axis=1)
    x_test_extend = np.concatenate((x_test_extend, X_test**k), axis=1)

    # Train the model using the training sets
    linear_regressor.fit(x_train_extend,y_train)

    # Record training MSE
    MSE_train.append(mean_squared_error(y_train, linear_regressor.predict(x_train_extend)))

    # Record test MSE
    MSE_test.append(mean_squared_error(y_test, linear_regressor.predict(x_test_extend)))
```

In [139]:
```
import matplotlib.pyplot as plt
# plot the train and test MSE values for various orders of K
# select model with minimum test MSE
fig, ax1 = plt.subplots(figsize=(12, 9))
plt.plot(k_value, MSE_train,  color = 'blue', marker = '.', markersize = 8, markeredgecolor = 'black', markerfacecolor = 'black',labe
plt.plot(k_value, MSE_test,  color = 'red',marker = '.', markersize = 8, markeredgecolor = 'black', markerfacecolor = 'black', label
plt.legend(fontsize=18)
plt.title('MSE with different values of k', fontsize=22)
plt.tick_params(labelsize=18)
plt.show()
```

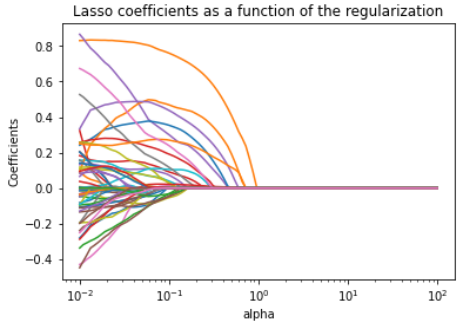


In [140]:
```
MSE_test
```

Out[140]:
```
[24.864087430472612,
 116.19868257792383,
 356.0625573143482,
 2667.9450042119006,
 6369.8434109012,
 247.4261623771244,
 255.26853107483294,
 344.2392877651073]
```

Model Evaluation

In [141]:
```
alphas = 10**np.linspace(-2,2,100)

lasso = Lasso()
coefs = []

for a in alphas:
    lasso.set_params(alpha=a, max_iter=10000) # increase iterations for optimization of coefficients
    lasso.fit(scale(X_train), y_train)
    coefs.append(lasso.coef_)

ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
ax.set_xlim(ax.get_xlim())
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('Coefficients')
# plt.legend(list(X_train.columns), loc='best')

plt.title('Lasso coefficients as a function of the regularization');
```



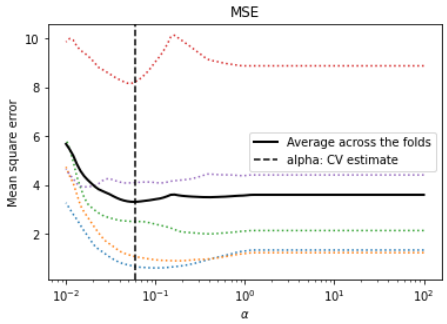
In [142]:
```
lassocv = LassoCV(alphas=alphas, max_iter=10000)
lassocv.fit(scale(X_train), y_train)

print('The best alpha from LassoCV:', lassocv.alpha_)
```

```
The best alpha from LassoCV: 0.05857020818056667
```

In [143]:
```
lasso.set_params(alpha=lassocv.alpha_)
lasso.fit(scale(X_train), y_train)
print('The coefficients are:')
print(pd.Series(lasso.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                             -0.089925
Total Equity Funding Amount Currency (in USD)         0.804784
Total Funding Amount Currency (in USD)                0.000000
Number of Investors                                   0.135621
Total Products Active                                 0.387561
Patents Granted                                      -0.000000
Trademarks Registered                                 0.000000
Founded Year                                          0.000000
Closed Year                                          -0.088807
Closed Month                                         -0.000000
CPI                                                  -0.000000
Unemployment                                          0.000000
Number of Employees_10001+                           -0.000000
Number of Employees_1001-5000                         0.000000
Number of Employees_101-250                           0.109402
Number of Employees_11-50                             0.000000
Number of Employees_251-500                          -0.000000
Number of Employees_5001-10000                       -0.034457
Number of Employees_501-1000                         -0.000000
Number of Employees_51-100                            0.068469
Last Funding Type_Grant                               0.377399
Last Funding Type_Post-IPO Equity                     0.497710
Last Funding Type_Private Equity                     -0.000000
Last Funding Type_Secondary Market                    0.234623
Last Funding Type_Seed                                0.000000
Last Funding Type_Series A                           -0.034034
Last Funding Type_Series B                            0.000000
Last Funding Type_Series C                           -0.000000
Last Funding Type_Series D                            0.115195
Last Funding Type_Series E                            0.000000
Last Funding Type_Series F                           -0.041254
Last Funding Type_Series G                           -0.000000
Last Funding Type_Venture - Series Unknown           -0.102041
Last Equity Funding Type_Private Equity              -0.000000
Last Equity Funding Type_Seed                         0.000000
Last Equity Funding Type_Series A                    -0.000000
Last Equity Funding Type_Series B                     0.000000
Last Equity Funding Type_Series C                     0.000000
Last Equity Funding Type_Series D                     0.000511
Last Equity Funding Type_Series E                     0.108083
Last Equity Funding Type_Series F                    -0.002709
Last Equity Funding Type_Series G                    -0.000000
Last Equity Funding Type_Venture - Series Unknown    -0.000000
IPO Status_Private                                   -0.000000
Estimated Revenue Range_$10B+                         0.485986
Estimated Revenue Range_$10M to $50M                  0.000000
Estimated Revenue Range_$1M to $10M                   0.000000
Estimated Revenue Range_$500M to $1B                  0.183701
Estimated Revenue Range_$50M to $100M                -0.001303
Estimated Revenue Range_Less than $1M                -0.000000
Estimated Revenue Range_N/A                          -0.000000
Founded Month_2.0                                     0.271050
Founded Month_3.0                                     0.000000
Founded Month_4.0                                    -0.000000
Founded Month_5.0                                     0.000000
Founded Month_6.0                                    -0.031027
Founded Month_7.0                                    -0.000000
Founded Month_8.0                                    -0.024481
Founded Month_9.0                                     0.070870
Founded Month_10.0                                    0.000000
Founded Month_11.0                                   -0.000000
Founded Month_12.0                                   -0.000000
State_CO                                             -0.000000
State_CT                                              0.020254
State_GA                                              0.000000
State_MA                                             -0.079740
State_MD                                              0.299468
State_ME                                              0.000000
State_MN                                             -0.000000
State_NC                                             -0.013681
State_NJ                                              0.000000
State_NV                                              0.000000
State_NY                                              0.000000
State_OR                                              0.000000
State_PA                                             -0.000000
State_TX                                             -0.098584
State_VA                                             -0.000000
State_WA                                             -0.000000
Industry_biotech                                      0.013245
Industry_enterprise                                  -0.000000
Industry_games_video                                  0.002173
Industry_hardware                                    -0.000000
Industry_mobile                                      -0.046853
Industry_other                                        0.008929
Industry_semiconductor                                0.000000
Industry_software                                    -0.000000
Industry_web                                         -0.009783
dtype: float64
```

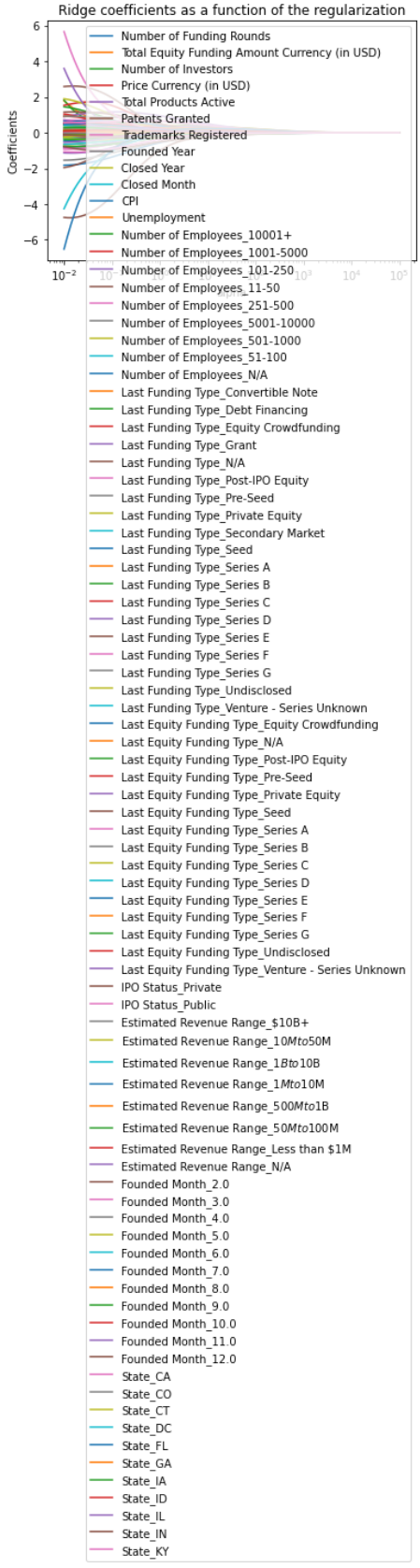
In [144]:
```
plt.semilogx(lassocv.alphas_, lassocv.mse_path_, linestyle=":")
plt.plot(
    lassocv.alphas_,
    lassocv.mse_path_.mean(axis=-1),
    color="black",
    label="Average across the folds",
    linewidth=2,
)
plt.axvline(lassocv.alpha_, linestyle="--", color="black", label="alpha: CV estimate")


plt.xlabel(r"$\alpha$")
plt.ylabel("Mean square error")
plt.legend()
plt.title("MSE")
plt.show()
```



In [145]:
```
lasso.set_params(alpha=lassocv.alpha_)
y_pred=lasso.predict(scale(X_test.values))
mse = round(mean_squared_error(y_test, y_pred),3)
print(f"MSE is {mse}")
```

```
MSE is 2.875
```

In [146]:
```
# ridge regression model
alphas = 10**np.linspace(-2,5,100)

ridge = Ridge()
coefs = []

for a in alphas:
    ridge.set_params(alpha=a)
    ridge.fit(scale(X_train), y_train)
    coefs.append(ridge.coef_)

ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
ax.set_xlim(ax.get_xlim())
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('Coefficients')
plt.legend(list(X.columns), loc='best')

plt.title('Ridge coefficients as a function of the regularization');
```



In [147]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler().fit(X_train)
```

In [148]:
```
# set a large alpha to get smaller coefficients
ridge = Ridge(alpha=1000)
ridge.fit(scaler.transform(X_train), y_train)

print('The coefficients are:')
print(pd.Series(ridge.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                              -0.003685
Total Equity Funding Amount Currency (in USD)          0.058473
Total Funding Amount Currency (in USD)                 0.057783
Number of Investors                                    0.022248
Total Products Active                                  0.031315
Patents Granted                                       -0.003367
Trademarks Registered                                  0.033235
Founded Year                                           0.004238
Closed Year                                           -0.016218
Closed Month                                          -0.007417
CPI                                                   -0.000858
Unemployment                                          -0.002561
Number of Employees_10001+                            -0.007117
Number of Employees_1001-5000                          0.025918
Number of Employees_101-250                            0.012679
Number of Employees_11-50                             -0.013501
Number of Employees_251-500                            0.003965
Number of Employees_5001-10000                        -0.008409
Number of Employees_501-1000                           0.025420
Number of Employees_51-100                             0.004351
Last Funding Type_Grant                                0.031071
Last Funding Type_Post-IPO Equity                      0.041727
Last Funding Type_Private Equity                      -0.004304
Last Funding Type_Secondary Market                     0.021595
Last Funding Type_Seed                                 0.004227
Last Funding Type_Series A                            -0.000211
Last Funding Type_Series B                            -0.010692
Last Funding Type_Series C                            -0.009808
Last Funding Type_Series D                            -0.001036
Last Funding Type_Series E                             0.007562
Last Funding Type_Series F                            -0.008949
Last Funding Type_Series G                            -0.001597
Last Funding Type_Venture - Series Unknown            -0.006451
Last Equity Funding Type_Private Equity               -0.004304
Last Equity Funding Type_Seed                          0.004227
Last Equity Funding Type_Series A                     -0.002386
Last Equity Funding Type_Series B                     -0.010692
Last Equity Funding Type_Series C                      0.002647
Last Equity Funding Type_Series D                     -0.001036
Last Equity Funding Type_Series E                      0.016733
Last Equity Funding Type_Series F                     -0.008949
Last Equity Funding Type_Series G                     -0.001597
Last Equity Funding Type_Venture - Series Unknown     -0.006451
IPO Status_Private                                    -0.023417
Estimated Revenue Range_$10B+                          0.040196
Estimated Revenue Range_$10M to $50M                   0.018657
Estimated Revenue Range_$1M to $10M                   -0.002761
Estimated Revenue Range_$500M to $1B                   0.013091
Estimated Revenue Range_$50M to $100M                 -0.000266
Estimated Revenue Range_Less than $1M                 -0.008884
Estimated Revenue Range_N/A                           -0.028935
Founded Month_2.0                                      0.051082
Founded Month_3.0                                     -0.000228
Founded Month_4.0                                     -0.009827
Founded Month_5.0                                     -0.005238
Founded Month_6.0                                     -0.006928
Founded Month_7.0                                     -0.009860
Founded Month_8.0                                     -0.009639
Founded Month_9.0                                      0.005031
Founded Month_10.0                                    -0.004182
Founded Month_11.0                                    -0.007914
Founded Month_12.0                                    -0.008884
State_CO                                              -0.008119
State_CT                                               0.041727
State_GA                                              -0.004691
State_MA                                              -0.017181
State_MD                                               0.020634
State_ME                                               0.000000
State_MN                                              -0.007134
State_NC                                              -0.004342
State_NJ                                               0.000000
State_NV                                               0.000000
State_NY                                              -0.000627
State_OR                                               0.000000
State_PA                                               0.001791
State_TX                                              -0.010230
State_VA                                              -0.005306
State_WA                                              -0.008456
Industry_biotech                                      -0.001377
Industry_enterprise                                   -0.001966
Industry_games_video                                   0.049676
Industry_hardware                                     -0.009226
Industry_mobile                                       -0.017923
Industry_other                                         0.017028
Industry_semiconductor                                -0.006762
Industry_software                                     -0.008101
Industry_web                                          -0.011126
dtype: float64
```

In [149]:
```
ridgecv = RidgeCV(alphas=alphas, scoring='neg_mean_squared_error')
ridgecv.fit(scale(X_train), y_train)

print('The best alpha from RidgeCV:', ridgecv.alpha_)
```

```
The best alpha from RidgeCV: 126.1856883066021
```

In [150]:
```
ridge.set_params(alpha=ridgecv.alpha_)
ridge.fit(scale(X_train), y_train)
print('The coefficients are:')
print(pd.Series(ridge.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                            -0.053750
Total Equity Funding Amount Currency (in USD)        0.199715
Total Funding Amount Currency (in USD)               0.196236
Number of Investors                                  0.073194
Total Products Active                                0.137511
Patents Granted                                     -0.044984
Trademarks Registered                                0.087638
Founded Year                                         0.014652
Closed Year                                         -0.057863
Closed Month                                        -0.024632
CPI                                                 -0.016127
Unemployment                                        -0.006094
Number of Employees_10001+                          -0.026642
Number of Employees_1001-5000                        0.072875
Number of Employees_101-250                          0.070748
Number of Employees_11-50                           -0.024451
Number of Employees_251-500                          0.014608
Number of Employees_5001-10000                      -0.033850
Number of Employees_501-1000                         0.054720
Number of Employees_51-100                           0.026428
Last Funding Type_Grant                              0.149231
Last Funding Type_Post-IPO Equity                    0.135180
Last Funding Type_Private Equity                    -0.022229
Last Funding Type_Secondary Market                   0.108146
Last Funding Type_Seed                               0.000280
Last Funding Type_Series A                          -0.023738
Last Funding Type_Series B                          -0.022696
Last Funding Type_Series C                          -0.047410
Last Funding Type_Series D                           0.016402
Last Funding Type_Series E                           0.021067
Last Funding Type_Series F                          -0.033983
Last Funding Type_Series G                          -0.011118
Last Funding Type_Venture - Series Unknown          -0.035152
Last Equity Funding Type_Private Equity             -0.022229
Last Equity Funding Type_Seed                        0.000280
Last Equity Funding Type_Series A                   -0.027571
Last Equity Funding Type_Series B                   -0.022696
Last Equity Funding Type_Series C                    0.020365
Last Equity Funding Type_Series D                    0.016402
Last Equity Funding Type_Series E                    0.068663
Last Equity Funding Type_Series F                   -0.033983
Last Equity Funding Type_Series G                   -0.011118
Last Equity Funding Type_Venture - Series Unknown   -0.035152
IPO Status_Private                                  -0.066474
Estimated Revenue Range_$10B+                        0.162305
Estimated Revenue Range_$10M to $50M                 0.050449
Estimated Revenue Range_$1M to $10M                 -0.015405
Estimated Revenue Range_$500M to $1B                 0.073504
Estimated Revenue Range_$50M to $100M               -0.018548
Estimated Revenue Range_Less than $1M               -0.018508
Estimated Revenue Range_N/A                         -0.074325
Founded Month_2.0                                    0.177430
Founded Month_3.0                                    0.003634
Founded Month_4.0                                   -0.025698
Founded Month_5.0                                   -0.013925
Founded Month_6.0                                   -0.025014
Founded Month_7.0                                   -0.059552
Founded Month_8.0                                   -0.031123
Founded Month_9.0                                    0.035322
Founded Month_10.0                                  -0.009369
Founded Month_11.0                                  -0.033640
Founded Month_12.0                                  -0.018508
State_CO                                            -0.030252
State_CT                                             0.135180
State_GA                                            -0.031535
State_MA                                            -0.070151
State_MD                                             0.121517
State_ME                                             0.000000
State_MN                                            -0.031172
State_NC                                            -0.019250
State_NJ                                             0.000000
State_NV                                             0.000000
State_NY                                             0.002837
State_OR                                             0.000000
State_PA                                            -0.003540
State_TX                                            -0.072752
State_VA                                            -0.019449
State_WA                                            -0.020373
Industry_biotech                                     0.009554
Industry_enterprise                                  0.001049
Industry_games_video                                 0.160924
Industry_hardware                                   -0.025855
Industry_mobile                                     -0.056084
Industry_other                                       0.052776
Industry_semiconductor                              -0.009689
Industry_software                                   -0.030426
Industry_web                                        -0.032905
dtype: float64
```

In [151]:
```
y_pred=ridge.predict(scale(X_test.values))
mse = round(mean_squared_error(y_test, y_pred),3)
print(f"MSE is {mse}")
```

```
MSE is 2.406
```

In [152]:
```
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import ElasticNetCV

ENcv = ElasticNetCV(alphas=None, cv=10, max_iter=10000) # default l1_ratio=0.5
ENcv.fit(scale(X_train), y_train)

print('The best alpha from ElasticNetCV:', ENcv.alpha_)
```

```
The best alpha from ElasticNetCV: 0.13532197256202563
```

In [153]:
```
# with the best alpha
EN=ElasticNet()
EN.set_params(alpha=ENcv.alpha_)
EN.fit(scale(X_train), y_train)

print('The coefficients are:')
print(pd.Series(EN.coef_.flatten(), index=X_train.columns))
```

```
The coefficients are:
Number of Funding Rounds                              -0.057446
Total Equity Funding Amount Currency (in USD)          0.404716
Total Funding Amount Currency (in USD)                 0.314231
Number of Investors                                    0.121308
Total Products Active                                  0.331908
Patents Granted                                       -0.000000
Trademarks Registered                                  0.035525
Founded Year                                           0.000000
Closed Year                                           -0.081137
Closed Month                                          -0.000000
CPI                                                   -0.000000
Unemployment                                           0.000000
Number of Employees_10001+                            -0.000000
Number of Employees_1001-5000                          0.000000
Number of Employees_101-250                            0.101256
Number of Employees_11-50                              0.000000
Number of Employees_251-500                           -0.000000
Number of Employees_5001-10000                        -0.012862
Number of Employees_501-1000                          -0.000000
Number of Employees_51-100                             0.062702
Last Funding Type_Grant                                0.350799
Last Funding Type_Post-IPO Equity                      0.242420
Last Funding Type_Private Equity                      -0.000000
Last Funding Type_Secondary Market                     0.207055
Last Funding Type_Seed                                 0.000000
Last Funding Type_Series A                            -0.009696
Last Funding Type_Series B                             0.000000
Last Funding Type_Series C                            -0.000000
Last Funding Type_Series D                             0.036033
Last Funding Type_Series E                             0.000000
Last Funding Type_Series F                            -0.016066
Last Funding Type_Series G                            -0.000000
Last Funding Type_Venture - Series Unknown            -0.036806
Last Equity Funding Type_Private Equity               -0.000000
Last Equity Funding Type_Seed                          0.000000
Last Equity Funding Type_Series A                     -0.000000
Last Equity Funding Type_Series B                      0.000000
Last Equity Funding Type_Series C                      0.000000
Last Equity Funding Type_Series D                      0.036145
Last Equity Funding Type_Series E                      0.112398
Last Equity Funding Type_Series F                     -0.015967
Last Equity Funding Type_Series G                     -0.000000
Last Equity Funding Type_Venture - Series Unknown     -0.036626
IPO Status_Private                                    -0.000000
Estimated Revenue Range_$10B+                          0.405830
Estimated Revenue Range_$10M to $50M                   0.000000
Estimated Revenue Range_$1M to $10M                    0.000000
Estimated Revenue Range_$500M to $1B                   0.155029
Estimated Revenue Range_$50M to $100M                 -0.000000
Estimated Revenue Range_Less than $1M                 -0.000000
Estimated Revenue Range_N/A                           -0.000000
Founded Month_2.0                                      0.277311
Founded Month_3.0                                      0.000000
Founded Month_4.0                                     -0.000000
Founded Month_5.0                                      0.000000
Founded Month_6.0                                     -0.008371
Founded Month_7.0                                     -0.010530
Founded Month_8.0                                     -0.021984
Founded Month_9.0                                      0.055142
Founded Month_10.0                                     0.000000
Founded Month_11.0                                    -0.000000
Founded Month_12.0                                    -0.000000
State_CO                                              -0.000000
State_CT                                               0.241971
State_GA                                              -0.000000
State_MA                                              -0.069989
State_MD                                               0.264418
State_ME                                               0.000000
State_MN                                              -0.000000
State_NC                                              -0.000000
State_NJ                                               0.000000
State_NV                                               0.000000
State_NY                                               0.000000
State_OR                                               0.000000
State_PA                                               0.000000
State_TX                                              -0.074983
State_VA                                              -0.000000
State_WA                                              -0.000000
Industry_biotech                                       0.001588
Industry_enterprise                                   -0.000000
Industry_games_video                                   0.070687
Industry_hardware                                     -0.000000
Industry_mobile                                       -0.043198
Industry_other                                         0.017395
Industry_semiconductor                                 0.000000
Industry_software                                     -0.000000
Industry_web                                          -0.000000
dtype: float64
```

In [154]:
```
y_pred=EN.predict(scale(X_test.values))
#best_model = y_pred
mse = round(mean_squared_error(y_test, y_pred),3)
print(f"MSE is {mse}")
```

```
MSE is 2.704
```

Expected Value Framework

Expected Benefit = (Probability of Acquisition)*(Predicted Selling Price - Predicted Total Funding) + (1 - Probability of Acquisition)*(Predicted Total Funding)

If the expected benefit is above 0, then venture capitalists should invest.

In [155]:
```
# Obtain Classifier Predicted values for the test data
# Random Forest
# Preprocess the data

encoder = LabelEncoder()
Yrf = encoder.fit_transform(merged_df_test['Acquisition'])
Xrf = merged_df_test.drop(['Acquisition'], axis=1)

for col in Xrf.select_dtypes(include='object').columns:
    Xrf[col] = Xrf[col].astype('category')

Xrf = pd.get_dummies(data=Xrf, drop_first=True)
```

In [156]:
```
#obtain predicted values for RF
y_pred_rf_total=best_rd.predict(Xrf)
```

```
/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but Rando
mForestClassifier was fitted without feature names
  warnings.warn(
```

In [157]:
```
#obtain predicted probabilities for RF for Acquisition
y_prob_rf = best_rd.predict_proba(Xrf)
print('Predicted probabilities:', y_prob_rf)
```

```
Predicted probabilities: [[0.39420414 0.60579586]
 [0.4997774  0.5002226 ]
 [0.43329683 0.56670317]
 ...
 [0.4044684  0.5955316 ]
 [0.39056662 0.60943338]
 [0.43954789 0.56045211]]

/Users/stephaniepalanca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:443: UserWarning: X has feature names, but Rando
mForestClassifier was fitted without feature names
  warnings.warn(
```

In [158]:
```
ev_df = merged_df_test
```

In [159]:
```
# Use this to add pred acquisition and prob to data
ev_df['Y_Pred_Acquisition'] = y_pred_rf_total
ev_df['Prob_Acquisition'] = y_prob_rf[:,0]
```

In [160]:
```
ev_df
```

Out[160]:

	Number of Employees	Number of Funding Rounds	Last Funding Type	Last Equity Funding Type	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	IPO Status	Total Products Active	...	Founded Month	Closed Year	Closed Month	State	Industry	CPI	U
0	11-50	2.0	Series A	Series A	-0.124511	-0.120412	5.0	-0.232204	Private	29.0	...	8.0	0.0	0.0	NY	web	1.1	
1	501-1000	9.0	Private Equity	Private Equity	0.680874	1.693347	11.0	1.275339	Private	36.0	...	8.0	0.0	0.0	CA	other	5.3	
2	501-1000	10.0	Series F	Series F	6.960358	6.002566	33.0	-0.232204	Private	63.0	...	1.0	0.0	0.0	CA	web	2.8	
3	51-100	9.0	Equity Crowdfunding	Equity Crowdfunding	0.370019	0.318547	11.0	-0.232204	Private	0.0	...	1.0	0.0	0.0	CA	other	4.0	
4	101-250	9.0	Series B	Series B	0.373500	1.235571	30.0	-0.232204	Private	0.0	...	1.0	0.0	0.0	CA	games_video	1.6	
5	1001-5000	2.0	Venture - Series Unknown	Venture - Series Unknown	-0.133729	-0.128378	2.0	6.874782	Private	20.0	...	2.0	0.0	0.0	CA	games_video	3.1	

In [161]:
```
# obtain regression predicted values for funding -- regular MLR
# Define X and y
X_mlr = merged_df_test.drop(columns=['Total Funding Amount Currency (in USD)'], axis=1)
y_mlr = merged_df_test['Total Funding Amount Currency (in USD)']

for col in X_mlr.select_dtypes(include='object').columns:
    X_mlr[col] = X_mlr[col].astype('category')
X_mlr = pd.get_dummies(data=X_mlr, drop_first=True)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_mlr, y_mlr, test_size=0.3, random_state=200)


# Create LinearRegression object and fit to training data
linear_regressor_ev = LinearRegression()
linear_regressor_ev.fit(X_train, y_train)

# Predict y values for test data
y_pred_fund = linear_regressor_ev.predict(X_mlr)
```

In [162]:
```
ev_df['Y_Pred_Funding'] = y_pred_fund
```

In [163]:
```
ev_df
```

Out[163]:

	Number of Employees	Number of Funding Rounds	Last Funding Type	Last Equity Funding Type	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	IPO Status	Total Products Active	...	Closed Year	Closed Month	State	Industry	CPI	Unemploym
0	11-50	2.0	Series A	Series A	-0.124511	-0.120412	5.0	-0.232204	Private	29.0	...	0.0	0.0	NY	web	1.1	
1	501-1000	9.0	Private Equity	Private Equity	0.680874	1.693347	11.0	1.275339	Private	36.0	...	0.0	0.0	CA	other	5.3	
2	501-1000	10.0	Series F	Series F	6.960358	6.002566	33.0	-0.232204	Private	63.0	...	0.0	0.0	CA	web	2.8	
3	51-100	9.0	Equity Crowdfunding	Equity Crowdfunding	0.370019	0.318547	11.0	-0.232204	Private	0.0	...	0.0	0.0	CA	other	4.0	
4	101-250	9.0	Series B	Series B	0.373500	1.235571	30.0	-0.232204	Private	0.0	...	0.0	0.0	CA	games_video	1.6	
5	1001-5000	2.0	Venture - Series Unknown	Venture - Series Unknown	-0.133729	-0.128378	2.0	6.874782	Private	20.0	...	0.0	0.0	CA	games_video	3.1	

In [164]:
```
ev_subset = ev_df[ev_df['Acquisition'] == 'Not Acquired']
ev_subset
```

Out[164]:

	Number of Employees	Number of Funding Rounds	Last Funding Type	Last Equity Funding Type	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	IPO Status	Total Products Active	...	Closed Year	Closed Month	State	Industry	CPI	Unemployme
0	11-50	2.0	Series A	Series A	-0.124511	-0.120412	5.0	-0.232204	Private	29.0	...	0.0	0.0	NY	web	1.1	4
3	51-100	9.0	Equity Crowdfunding	Equity Crowdfunding	0.370019	0.318547	11.0	-0.232204	Private	0.0	...	0.0	0.0	CA	other	4.0	4
4	101-250	9.0	Series B	Series B	0.373500	1.235571	30.0	-0.232204	Private	0.0	...	0.0	0.0	CA	games_video	1.6	4
6	11-50	3.0	Series A	Series A	-0.033476	-0.041737	4.0	-0.232204	Private	0.0	...	0.0	0.0	CA	web	1.6	4
8	101-250	3.0	Venture - Series Unknown	Venture - Series Unknown	0.084333	0.060078	2.0	-0.232204	Private	34.0	...	0.0	0.0	CA	web	1.6	6

In [165]:
```
print(ev_subset['Y_Pred_Acquisition'].value_counts())
```

```
1    327
0     52
Name: Y_Pred_Acquisition, dtype: int64
```

In [166]:
```
#Obtain predicted values for price (for those that were acquired AND have price information)
df_acquired = merged_df_test[merged_df_test["Acquisition"] == "Acquired"]
df_price = df_acquired[df_acquired["Price Currency (in USD)"] > 0]

X1 = df_price.drop(columns=['Price Currency (in USD)'])
y1 = df_price['Price Currency (in USD)']

for col in X1.select_dtypes(include='object').columns:
    X1[col] = X1[col].astype('category')
X1 = pd.get_dummies(data=X1, drop_first=True)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3, random_state=200)
```

In [167]:
```
ridge = Ridge()
ridge.fit(X_train, y_train)

y_pred_price = ridge.predict(X1)
```

In [168]:
```
df_price
```

Out[168]:

	Number of Employees	Number of Funding Rounds	Last Funding Type	Last Equity Funding Type	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	IPO Status	Total Products Active	...	Closed Year	Closed Month	State	Industry	CPI	Unemployment	Acq
1	501-1000	9.0	Private Equity	Private Equity	0.680874	1.693347	11.0	1.275339	Private	36.0	...	0.0	0.0	CA	other	5.3	6.1	A
5	1001-5000	2.0	Venture - Series Unknown	Venture - Series Unknown	-0.133729	-0.128378	2.0	6.874782	Private	20.0	...	0.0	0.0	CA	games_video	3.1	5.4	A
27	51-100	4.0	Series C	Series C	-0.054896	-0.060249	6.0	0.004696	Private	70.0	...	0.0	0.0	CA	games_video	4.0	4.7	A
31	51-100	4.0	Grant	Series C	0.202141	0.162123	8.0	5.367239	Private	18.0	...	0.0	0.0	CA	other	2.1	4.6	A
36	1001-5000	6.0	Series C	Series C	-0.029996	-0.038729	5.0	0.413886	Private	39.0	...	0.0	0.0	TX	other	2.1	4.6	A
39	251-500	3.0	Series B	Series B	0.151269	0.117927	11.0	4.120157	Private	8.0	...	0.0	0.0	CA	other	1.2	9.4	A
				Post IPO	Post IPO													

In [169]:
```
#df_price_2 = df_price
df_price_2 = pd.DataFrame(df_price)
df_price_2['Y_Pred_Price'] = y_pred_price
```

In [170]:
```
len(df_price_2)
```

Out[170]: 111

In [171]:
```
df_price_2
```

Out[171]:

	Number of Employees	Number of Funding Rounds	Last Funding Type	Last Equity Funding Type	Total Equity Funding Amount Currency (in USD)	Total Funding Amount Currency (in USD)	Number of Investors	Price Currency (in USD)	IPO Status	Total Products Active	...	Closed Month	State	Industry	CPI	Unemployment	Acquisition
1	501-1000	9.0	Private Equity	Private Equity	0.680874	1.693347	11.0	1.275339	Private	36.0	...	0.0	CA	other	5.3	6.1	Acquired
5	1001-5000	2.0	Venture - Series Unknown	Venture - Series Unknown	-0.133729	-0.128378	2.0	6.874782	Private	20.0	...	0.0	CA	games_video	3.1	5.4	Acquired
27	51-100	4.0	Series C	Series C	-0.054896	-0.060249	6.0	0.004696	Private	70.0	...	0.0	CA	games_video	4.0	4.7	Acquired
31	51-100	4.0	Grant	Series C	0.202141	0.162123	8.0	5.367239	Private	18.0	...	0.0	CA	other	2.1	4.6	Acquired
36	1001-5000	6.0	Series C	Series C	-0.029996	-0.038729	5.0	0.413886	Private	39.0	...	0.0	TX	other	2.1	4.6	Acquired
39	251-500	3.0	Series B	Series B	0.151269	0.117927	11.0	4.120157	Private	8.0	...	0.0	CA	other	1.2	9.4	Acquired
				Post IPO	Post IPO												

In [172]:
```
#Expected Value
df_price_2['Expected_Value_1'] = df_price_2['Prob_Acquisition']*(df_price_2['Y_Pred_Price'] - df_price_2['Y_Pred_Funding'])
```

In [173]:
```
df_price_2['Expected_Value_2'] = (1-df_price_2['Prob_Acquisition'])*-df_price_2['Y_Pred_Funding']
```

In [174]:
```
df_price_2['Expected_Benefit'] = df_price_2['Expected_Value_1'] + df_price_2['Expected_Value_2']
```

In [175]:
```
df_price_2 = df_price_2.drop(columns = ['Expected_Value_1', 'Expected_Value_2'])
```

In [176]:
```
df_price_2.head()
```

Out[176]:

umber of estors	Price Currency (in USD)	IPO Status	Total Products Active	...	State	Industry	CPI	Unemployment	Acquisition	Y_Pred_Acquisition	Prob_Acquisition	Y_Pred_Funding	Y_Pred_Price	Expected_Benefit
11.0	1.275339	Private	36.0	...	CA	other	5.3	6.1	Acquired	1	0.499777	0.943309	1.024132	-0.431471
2.0	6.874782	Private	20.0	...	CA	games_video	3.1	5.4	Acquired	1	0.458057	-0.128378	5.623133	2.704091
6.0	0.004696	Private	70.0	...	CA	games_video	4.0	4.7	Acquired	0	0.559932	-0.058210	5.019167	2.868602
8.0	5.367239	Private	18.0	...	CA	other	2.1	4.6	Acquired	0	0.536169	0.068977	4.192366	2.178839
5.0	0.413886	Private	39.0	...	TX	other	2.1	4.6	Acquired	1	0.490350	-0.084815	1.963013	1.047379

In [177]:
```
df_price_2['Investment_Decision'] = df_price_2.apply(lambda row: 'Invest' if row['Expected_Benefit'] > 0 else 'Do Not Invest', axis=
```

In [178]:
```
df_price_2
```

10.0	0.982445	Private	0.0	...	mobile	2.0	5.7	Acquired	0	0.571104	-0.099242	-0.559029	-0.220022	Do Not Invest
9.0	2.718272	Private	6.0	...	other	0.0	8.3	Acquired	1	0.480208	0.563632	2.841900	0.801072	Invest
5.0	0.080073	Private	7.0	...	enterprise	1.7	5.8	Acquired	0	0.562259	-0.081338	0.973883	0.628913	Invest
11.0	2.998244	Private	7.0	...	other	4.9	5.6	Acquired	0	0.556404	0.040190	-0.608122	-0.378551	Do Not Invest
4.0	0.327740	Private	16.0	...	software	1.2	5.7	Acquired	0	0.533690	0.102333	1.128144	0.499745	Invest
6.0	0.736931	Private	4.0	...	other	2.8	4.0	Acquired	0	0.555191	-0.096374	1.274744	0.804100	Invest
8.0	0.556025	Private	5.0	...	advertising	3.1	5.4	Acquired	0	0.556274	0.003473	1.775052	0.983943	Invest
6.0	1.296875	Private	8.0	...	other	4.3	5.0	Acquired	1	0.499037	0.191851	1.860774	0.736743	Invest
4.0	0.866148	Private	2.0	...	mobile	2.0	5.7	Acquired	0	0.529646	0.200401	0.704966	0.172981	Invest
11.0	0.736931	Private	0.0	...	enterprise	2.8	4.0	Acquired	0	0.503432	0.433397	1.673235	0.408963	Invest
11.0	0.069305	Private	2.0	...	hardware	-2.0	9.5	Acquired	1	0.495761	-0.090502	0.726056	0.450452	Invest
5.0	0.844612	Private	1.0	...	software	1.2	5.7	Acquired	0	0.582474	-0.053546	1.082975	0.684351	Invest

In [179]:
```
print(df_price_2['Investment_Decision'].value_counts())
```

```
Invest           100
Do Not Invest     11
Name: Investment_Decision, dtype: int64
```

NOTE: Of the ones that are acquired (that we have price information for), we are only recommending that you invest in 91 of these. Just because a company was acquired, that does not mean it should be invested in. If given pricing information, we can build a larger dataset and a more accurate model.

Exporatory Data Analysis with the Probability of Acquisition, Predicted Returns, Predicted Funding Amount, and Expected Benefit

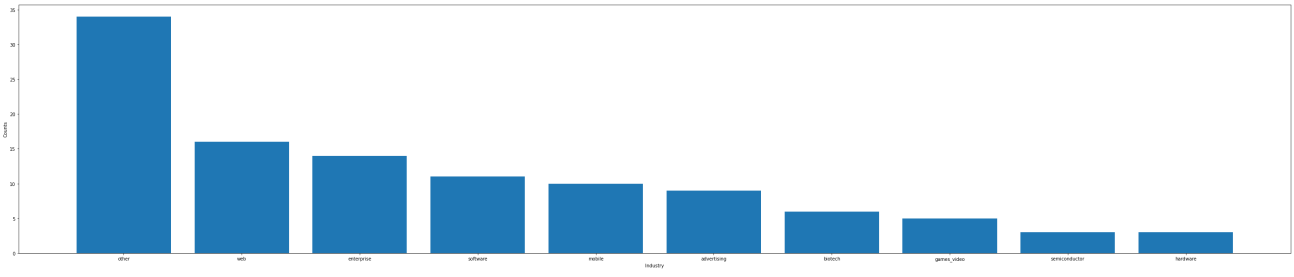
In [180]:
```
df_price_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 111 entries, 1 to 1039
Data columns (total 28 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Number of Employees                         111 non-null    object
 1   Number of Funding Rounds                    111 non-null    float64
 2   Last Funding Type                           111 non-null    object
 3   Last Equity Funding Type                    111 non-null    object
 4   Total Equity Funding Amount Currency (in USD)  111 non-null    float64
 5   Total Funding Amount Currency (in USD)      111 non-null    float64
 6   Number of Investors                         111 non-null    float64
 7   Price Currency (in USD)                     111 non-null    float64
 8   IPO Status                                  111 non-null    object
 9   Total Products Active                       111 non-null    float64
 10  Patents Granted                             111 non-null    float64
 11  Trademarks Registered                       111 non-null    float64
 12  Estimated Revenue Range                     111 non-null    object
 13  Founded Year                                111 non-null    float64
 14  Founded Month                               111 non-null    object
 15  Closed Year                                 111 non-null    float64
 16  Closed Month                                111 non-null    float64
 17  State                                       111 non-null    object
 18  Industry                                    111 non-null    object
 19  CPI                                         111 non-null    float64
 20  Unemployment                                111 non-null    float64
 21  Acquisition                                 111 non-null    object
 22  Y_Pred_Acquisition                          111 non-null    int64
 23  Prob_Acquisition                            111 non-null    float64
 24  Y_Pred_Funding                              111 non-null    float64
 25  Y_Pred_Price                                111 non-null    float64
 26  Expected_Benefit                            111 non-null    float64
 27  Investment_Decision                         111 non-null    object
dtypes: float64(17), int64(1), object(10)
memory usage: 25.1+ KB
```

In [181]:
```
df_invest = df_price_2[df_price_2['Investment_Decision'] == 'Invest']
```

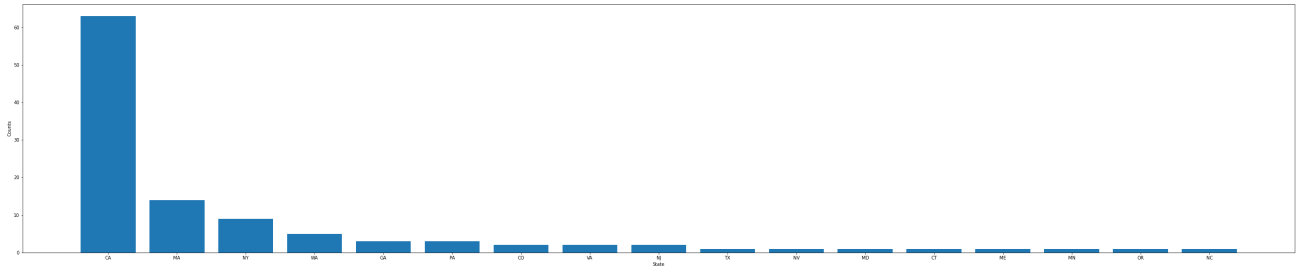
In [182]:
```
industry = df_price_2['Industry'].value_counts()
fig, ax = plt.subplots(figsize=(50, 10))
plt.bar(industry.index, industry.values)
plt.xlabel("Industry")
plt.ylabel("Counts")
```

Out[182]:
```
Text(0, 0.5, 'Counts')
```



In [183]:
```
state = df_price_2['State'].value_counts()
fig, ax = plt.subplots(figsize=(50, 10))
plt.bar(state.index, state.values)
plt.xlabel("State")
plt.ylabel("Counts")
```

Out[183]:
```
Text(0, 0.5, 'Counts')
```



In [184]:
```
fund_type = df_price_2['Last Funding Type'].value_counts()
fig, ax = plt.subplots(figsize=(50, 10))
plt.bar(fund_type.index, fund_type.values)
plt.xlabel("Last Funding Type")
plt.ylabel("Counts")
```

Out[184]:
```
Text(0, 0.5, 'Counts')
```