ensemblemethods_hw

April 15, 2025

Worked with: - Trevor Mathisen

- Viet Nguyen

1. Download startup failure dataset and its description.
2. Perform standard EDA to get familiar with the dataset.

[1]:
```
from sklearn.model_selection import KFold, cross_val_score
from numpy import set_printoptions
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import re
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier,␣
↪AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, roc_curve
```

[2]:
```
import os
dfs = []
for dirname, _, filenames in os.walk('datasets'):
    for filename in filenames:
        if '(' in filename and 'Food' not in filename:
            print(f'Loading {filename}...')
            df_temp = pd.read_csv(f'datasets/{filename}', encoding='utf-8')
            print(df_temp.shape)
            dfs.append(df_temp)

data = pd.concat(dfs, ignore_index=True)
set_printoptions(precision=3)
data = data.dropna()
print(data.isnull().sum())
print(data.shape)
# Display unique values in each column
for col in data.columns:
    unique_values = data[col].unique()
    print(f"Unique values in '{col}': {unique_values}")
```

```
Loading Startup Failure (Manufactures).csv…
(30, 20)
Loading Startup Failures (Information Sector).csv…
(156, 20)
Loading Startup Failure (Retail Trade).csv…
(90, 20)
Loading Startup Failure (Health Care).csv…
(60, 20)
Loading Startup Failure (Finance and Insurance).csv…
(47, 20)
Name                      0
Sector                    0
Years of Operation        0
What They Did             0
How Much They Raised      0
Why They Failed           0
Takeaway                  0
Giants                    0
No Budget                 0
Competition               0
Poor Market Fit           0
Acquisition Stagnation    0
Platform Dependency       0
Monetization Failure      0
Niche Limits              0
Execution Flaws           0
Trend Shifts              0
Toxicity/Trust Issues     0
Regulatory Pressure       0
Overhype                  0
dtype: int64
(382, 20)
Unique values in 'Name': ['Airware' 'Anki' 'Aptera Motors' 'Aria Insights'
'August Home'
 'BeagleBone' 'Better Place' 'Butterfly Network' 'Cubelets'
 "Dyson's EV Project" 'Elio Motors' 'Essential Products' 'Faraday Future'
 'Fisker Automotive' 'GoPro Karma' 'Hello' 'Jawbone' 'Juicero' 'Light'
 'Lily Robotics' 'Lytro' 'MakerBot' 'Osmo Systems' 'Pearl Automation'
 'Pebble' 'Quanergy' 'Rethink Robotics' 'Skully Helmets' 'Thalmic Labs'
 'Zume' 'Airy Labs' 'Ask Jeeves' 'Bebo' 'Burbn' 'Canvas' 'Change.org'
 'Chirp' 'Cloudera' 'Cocoon' 'Codeacademy' 'CollabFinder' 'Color Labs'
 'Connect' 'Coub' 'Domo' 'Digg' 'Drifty' 'Dropbox Paper' 'Evernote' 'Exec'
 'EyeEm' 'Factual' 'Formspring' 'Foursquare Swarm' 'Friendster'
 'GeoCities' 'GetGlue' 'Gigya' 'Gimlet' 'Gowalla' 'Grooveshark' 'GroupMe'
 'HootSuite Media' 'Houseparty' 'Huddle' 'Hulu Japan' 'IFTTT' 'Inkling'
 'Instagram Live' 'Intercom' 'Invision' 'Jetpac' 'Kik' 'Knewton' 'Mailbox'
 'Maker Media' 'Medium One' 'Milk' 'Mixpanel' 'Mobli' 'Music.ly' 'MySpace'
 'Netscape' 'NewsTilt' 'Notion AI' 'Nuzzel' 'Path' 'Periscope'
```

'Photobucket' 'Pinterest UK' 'Piston Cloud' 'Plex' 'Pocket' 'Pulse'
'Quibb' 'Quid' 'Quixey' 'Rap Genius' 'Readability' 'Refresh' 'RethinkDB'
'Riffsy' 'Rockmelt' 'Secret' 'ShareThis' 'Songza' 'SoundCloud Go'
'Soundwave' 'Spoke' 'Substack Local' 'Tango' 'Tasty Labs' 'TinyChat'
'Tinder Social' 'TokBox' 'Top Hat' 'Topix' 'Trello Gold' 'Tumblr'
'Treehouse' 'TuneIn Premium' 'Twilio SendGrid' 'Twitter Fabric'
'Twitter Music' 'Udacity Blitz' 'Vdio' 'Vine' 'VisiCalc' 'Whisper'
'WhoSay' 'Wickr' 'Wishbone' 'Woo' 'Workable' 'Xmarks' 'Xobni'
'Yahoo Answers' 'Yahoo Buzz' 'Yahoo Groups' 'Yahoo Live' 'Yammer'
'Yik Yak' 'Yo' 'Yola' 'YouNow' 'Zapier Plus' 'Zencoder' 'Zendesk Chat'
'Ziddu' 'Zombie Labs' 'Zoomdata' 'Zopim' 'Zscaler Shift' 'Zulily'
'Zumper Pro' 'Zurb' 'Zuuka' 'Zynga Games' 'Bitpass' 'Blync' 'BountyJobs'
'BranchOut' 'Burstly' 'CloudHammer' 'Crunchyroll' 'Curse' 'DailyBurn'
'Dailylook' 'Datto' 'Deem' 'Detour' 'Disruptive Media' 'Divshot'
'Drawbridge' 'Dunwello' 'Everalbum' 'Friend.ly' 'Frontback' 'Froyo'
'GoAnimate' 'Hitpost' 'Homer' 'Hot Potato' 'Humanoid' '99dresses'
'Ahalife' 'AllRomance' 'Auctionata' 'Augury Books' 'Beepi' 'Boxed'
'BurstIQ' 'Carwoo' 'Catelyn' 'Chinictown' 'Combatant Gentlemen'
'ContextLogic' 'Crate' 'Din' 'Dot & Bo' 'Drizly' 'Drync' 'Ecomom'
'ElectricObjects' 'Fab.com' 'Fancy' 'Fashism' 'Fiksu' 'Fobo' 'Gilt Taste'
'Goldbely' 'Good Eggs' 'Graze' 'Groupon Now' 'Havenly' 'Heyday' 'HobbyDB'
'Homejoy' 'Hush' 'Ibotta' 'Imercive' 'Incredibowl' 'Ipsy' 'Ista'
'JustFab' 'Karma' 'Kozmo.com' 'Lot18' 'Markafoni' 'Massdrop' 'Mayvenn'
'Memebox' 'ModCloth' 'MoveLoot' 'Nasty Gal' 'One Kings Lane' 'Operator'
'Outdoor Voices' 'Peerby' 'PetCube' 'Pets.com' 'Plum' 'Plum District'
'PopSugar Shop' 'Poshmark UK' 'Quirky' 'Raise' 'Rent the Runway UK'
'RetailMeNot UK' 'Sampling Lab' 'Selltag' 'ShopKeep' 'Shopkick'
'Shoptiques' 'Slice' 'Spring' 'Teespring' 'ThredUp Goody'
'ThriftBooks Indie' 'Thrillist Rewards' 'Thrive Market Organic'
'ToVieFor' 'Tradesy' 'Trunk Club' 'Try.com' 'Webvan' 'Wittlebee' 'Woot'
'Yelp Deals' 'Zaarly' 'Zappos Labs' 'Zappos Local' 'Zing' 'Zola Books'
'Aira Health' 'Amino' 'Arivale' 'Augmedix' 'Avizia' 'Babylon Health'
'Basis' 'Better' 'BetterHelp' 'BioBeats' 'Call9' 'Cardiogram' 'CareSync'
'CareZone' 'Clarityn' 'Clinkle' 'Cue Health' 'Dopamine Labs' 'Doxy.me'
'Driver' 'Dthera Sciences' 'Eargo' 'FitStar' 'Ginger' 'Goldfinch Bio'
'Healx' 'Health IQ' 'HealthRiser' 'HealthSpot' 'HealthTap+' 'HealthifyMe'
'InTouch Health' 'iRhythm' 'Kaiku Health' 'Kyruus' 'Lantern'
'Lantern Pharma' 'Luminary Labs' 'Lumeon' 'MediChain' 'Medicasafe'
'Mindstrong' 'Modern Health' 'Nurx' 'Olive' 'Omada Health'
'Outset Medical' 'Pear Therapeutics' 'PillPack' 'Proteus Digital Health'
'Quit Genius' 'Scanadu' 'Sherpaa' 'Sprig' 'StethoCloud' 'Theranos'
'Tinnitracks' 'uBiome' 'Zeo' 'Avant' 'Cake Financial' 'Circle'
'Clarity Money' 'Coinbase NFT' 'FundersClub' 'Fuze Network'
'Indiegogo Life' 'Isentium' 'LendUp' 'LendingClub' 'LendLayer' 'Loyal3'
'Money360' 'Monitor110' 'Mozido' 'Pawngo' 'Pay By Touch' 'Plum Will'
'PoundPay' 'ReadyForZero' 'RushCard' 'Seed' 'Sensible' 'SigFig' 'Simple'
'SmartAsset' 'SoFi Social' 'Square Cash' 'Taulia' 'Tilt' 'Toshl'
'TrustEgg' 'Upstart' 'Vemo' 'Venmo Groups' 'Vittana' 'Wealthfront Cash'

```
 'Wepay' 'Wesabe' 'Wise' 'YayPay' 'Ycharts' 'ZestFinance']
Unique values in 'Sector': ['Manufacturing' 'Information' 'Retail Trade' 'Health
Care'
 'Finance and Insurance']
Unique values in 'Years of Operation': ['2011-2018' '2010-2019' '2005-2011'
'2008-2019' '2012-2017' '2007-2013'
 '2011-2020' '2017-2019' '2009-2020' '2015-2020' '2014-2023' '2015-2018'
 '2006-2017' '2013-2017' '2013-2020' '2006-2018' '2009-2017' '2014-2017'
 '2012-2016' '2012-2022' '2008-2018' '2013-2016' '2012-2020' '2015-2023'
 '2 (2010-2012)' '11 (1996-2007)' '14 (2005-2019)' '6 (2011-2017)'
 '15 (2007-2022)' '5 (2011-2016)' '15 (2008-2023)' '4 (2018-2022)'
 '11 (2011-2022)' '7 (2011-2018)' '3 (2011-2014)' '6 (2010-2016)'
 '9 (2012-2021)' '13 (2010-2023)' '14 (2004-2018)' '7 (2012-2019)'
 '6 (2015-2021)' '15 (2004-2019)' '3 (2012-2015)' '12 (2011-2023)'
 '14 (2008-2022)' '7 (2009-2016)' '7 (2014-2021)' '12 (2002-2014)'
 '11 (1994-2005)' '8 (2008-2016)' '12 (2006-2018)' '5 (2007-2012)'
 '11 (2006-2017)' '5 (2010-2015)' '5 (2016-2021)' '8 (2011-2019)'
 '12 (2010-2022)' '11 (2009-2020)' '18 (1999-2017)' '13 (2009-2022)'
 '12 (2008-2020)' '6 (2014-2020)' '14 (2009-2023)' '7 (2010-2017)'
 '4 (2014-2018)' '15 (2003-2018)' '9 (1994-2003)' '5 (2018-2023)'
 '8 (2010-2018)' '5 (2015-2020)' '8 (2012-2020)' '11 (2007-2018)'
 '6 (2013-2019)' '8 (2009-2017)' '6 (2009-2015)' '10 (2009-2019)'
 '5 (2009-2014)' '13 (2007-2020)' '10 (2007-2017)' '6 (2016-2022)'
 '4 (2012-2016)' '3 (2020-2023)' '9 (2009-2018)' '5 (2017-2022)'
 '8 (2015-2023)' '2 (2013-2015)' '5 (2012-2017)' '5 (1979-1984)'
 '9 (2010-2019)' '5 (2013-2018)' '10 (2012-2022)' '8 (2006-2014)'
 '7 (2006-2013)' '15 (2005-2020)' '4 (2008-2012)' '15 (1998-2013)'
 '2 (2008-2010)' '10 (2008-2018)' '12 (2007-2019)' '7 (2015-2022)'
 '6 (2002-2008)' '15 (2006-2021)' '6 (2008-2014)' '14 (2006-2020)'
 '9 (2011-2020)' '14 (2000-2014)' '10 (2011-2021)' '3 (2010-2013)'
 '3 (2013-2016)' '14 (2007-2021)' '4 (2009-2013)' '2 (2009-2011)'
 '10 (2006-2016)' '6 (2012-2018)' '4 (2013-2017)' '8 (2013-2021)'
 '3 (2015-2018)' '11 (2012-2023)' '11 (2008-2019)' '4 (2011-2015)'
 '9 (2014-2023)' '8 (2014-2022)' '4 (2016-2020)' '4 (1998-2002)'
 '9 (2013-2022)' '15 (2002-2017)' '2 (1998-2000)' '11 (2010-2021)'
 '3 (1996-1999)' '12 (2004-2016)' '2015-2019' '2013-2021' '2012-2024'
 '2014-2018' '2013-2023' '2014-2016' '2016-2023' '2014-2020' '2016-2021'
 '2016-2020' '2011-2016' '2014-2024' '2013-2018' '2018-2018' '2010-2023'
 '2013-2015' '2011-2021' '2010-2017' '2002-2020' '2006-2015' '2010-2015'
 '2012-2018' '2017-2021' '2005-2023' '2017-2020' '2017-2023' '2014-2019'
 '2012-2023' '2004-2020' '2012-2019' '2012-2015' '2003-2018' '2008-2013'
 '2002-2008' '2006-2011' '2016-2022' '2021-2023' '2010-2016' '2012-2021'
 '2006-2021' '2005-2008' '2008-2017' '2011-2017' '2002-2007' '2009-2013'
 '2007-2018' '2009-2016' '2009-2022' '2008-2020' '2015-2021' '2008-2015'
 '2018-2023' '2011-2023' '2009-2021']
Unique values in 'What They Did': ['Drone hardware/software for industry' 'AI-
powered toy robots'
 'Three-wheeled electric vehicles' 'Tethered industrial drones'
```

'Smart locks and doorbells' 'Open-source computers'
'EVs with swappable batteries' 'Handheld ultrasound device'
'Modular toy robots' 'Electric vehicle' 'Cheap three-wheeled car'
'Premium smartphones' 'Luxury electric vehicles'
'Luxury hybrid-electric cars' 'Consumer drone' 'Sleep-tracking orb'
'Fitness trackers/speakers' 'Wi-Fi juicer' 'Multi-lens camera'
'Self-flying drone camera' 'Light-field cameras' 'Desktop 3D printers'
'IoT water sensors' 'Car rearview cameras' 'Smartwatches'
'LiDAR for autonomous vehicles' 'Collaborative robots'
'AR motorcycle helmets' 'Gesture-control armband' 'Robotic pizza systems'
'Educational mobile games for kids'
'Early search engine with butler mascot'
'Social networking site popular in UK' 'Check-in app with photo-sharing'
'Collaborative document editing platform'
'Petition platform for activism' 'App to share data via sound'
'Big data analytics platform' 'Privacy-focused browser'
'Online coding lessons' 'Matchmaking for project collaborators'
'Photo-sharing by location' 'Social networking via Facebook'
'Short looping video app' 'Business intelligence dashboards'
'Social news aggregator' 'Ionic mobile app framework'
'Collaborative doc tool' 'Note-taking app'
'On-demand personal assistants' 'Photo-sharing and stock platform'
'Location data platform' 'Anonymous Q&A social app'
'Check-in app with badges' 'Early social network'
'Web hosting for personal pages' 'Social TV check-in app'
'Social login identity tools' 'Podcasting narrative shows'
'Location-based social app' 'Music streaming with uploads'
'Group messaging app' 'Social media management tool'
'Group video chat app' 'Cloud collaboration for teams'
'Streaming for Japan' 'Automation connecting apps'
'Digital publishing for e-books' 'Live-streaming on Instagram'
'Customer messaging platform' 'Design collaboration platform'
'Fitness trackers with data' 'AI travel recommendation app'
'Messaging app for teens' 'Adaptive learning platform' 'Sleek email app'
'DIY tech magazine and events' 'IoT data processing platform'
'Mobile apps like Oink' 'User behavior analytics'
'Photo and video-sharing app' 'Lip-sync video app'
'Social networking pioneer' 'First big web browser'
'Platform for journalists' 'AI productivity feature'
'News curation via Twitter' 'Private social network' 'Live-streaming app'
'Photo-hosting site' 'UK-specific curation' 'OpenStack cloud software'
'Media server streaming' 'Save-for-later app' 'News reader app'
'Professional networking links' 'AI trend analytics'
'Mobile app search engine' 'Lyrics annotation site'
'Reading app with subscriptions' 'Social CRM app' 'Open-source database'
'GIF search engine' 'Social browser' 'Anonymous social app'
'Social sharing widget' 'Mood-based music streaming'
'Paid streaming tier' 'Social music tracking' 'AI workplace chatbot'

'Local newsletter funding' 'Video chat with games' 'Social Q&A app'
'Video chat rooms' 'Group hangout feature' 'Video chat platform'
'Classroom engagement tools' 'Local news aggregator'
'Premium project management' 'Microblogging platform'
'Online coding school' 'Paid radio streaming' 'Email API service'
'Developer toolkit' 'Music discovery app' 'Freelance gig platform'
'Video streaming service' 'Short-video app' 'First spreadsheet software'
'Celeb social platform' 'Secure messaging app' 'Polling app for teens'
'Dating app for pros' 'HR hiring software' 'Bookmark sync tool'
'Email plugin' 'Q&A platform' 'Online community forums'
'Live-streaming platform' 'Enterprise social network'
'Anonymous local social app' 'Minimalist messaging app' 'Website builder'
'Premium automation tier' 'Video encoding service' 'Live chat tool'
'File-sharing platform' 'Zombie-themed mobile games'
'Data visualization tool' 'Live chat widget' 'Cloud security analytics'
'Flash sales platform' 'Predictive pizza analytics'
'Rental listing premium' 'Design prototyping tool'
"Digital kids' storytelling" 'Social gaming giant'
'Micropayment platform' 'Social polling app' 'Recruitment marketplace'
'LinkedIn on Facebook' 'Mobile ad platform' 'Cloud management tool'
'Anime streaming platform' 'Gaming community platform'
'Fitness video app' 'Fashion subscription boxes'
'Backup and recovery software' 'Travel management platform'
'Audio tour app' 'Digital content platform' 'Web hosting builder'
'Ad tech identity tracking' 'Freelancer review platform'
'Photo storage app' 'Dual-camera photo app' 'Social gaming app'
'DIY animation tool' 'Sports social app' "Kids' reading app"
'Social check-in app' 'Content curation platform' 'Fashion swapping app'
'Curated luxury goods marketplace' 'E-book retailer for romance novels'
'Online auction house for art and luxury'
'Indie e-commerce bookstore for poetry'
'Peer-to-peer used car marketplace' 'Bulk e-commerce platform'
'Blockchain health data marketplace' 'Online car-buying platform'
'Subscription jewelry rental service' 'E-commerce for Chinese gadgets'
'Online menswear brand' 'Discount e-commerce app (Wish)'
'Online furniture rental service' 'Curated online grocery platform'
'Online home decor retailer' 'Alcohol delivery app'
'Wine e-commerce and discovery app'
'Eco-friendly baby products e-commerce' 'Digital art screens retailer'
'Flash-sale design retailer' 'Social e-commerce platform'
'Fashion advice app' 'Mobile marketing for retail apps'
'Local food surplus marketplace' 'Gourmet food e-commerce'
'Gourmet food delivery service' 'Online grocery delivery'
'Snack subscription box' 'Real-time deal app'
'Online interior design service' 'Skincare e-commerce and spas'
'Collectibles marketplace' 'On-demand cleaning service'
'Sleep products e-commerce' 'Cashback rewards app'
'Interactive shopping platform' 'Smoking accessories e-commerce'

'Beauty subscription box' 'Fashion rental marketplace'
'Fashion subscription service' 'Mobile deals marketplace'
'Urban free delivery service' 'Wine flash-sale site'
'Fashion e-commerce in Turkey' 'Group-buy electronics retailer'
'Hair extension e-commerce' 'Korean beauty subscription'
'Indie fashion e-commerce' 'Used furniture marketplace'
'Fast-fashion e-commerce' 'Home decor e-commerce'
'AI shopping assistant app' 'Activewear e-commerce'
'Peer-to-peer rental marketplace' 'Pet tech e-commerce'
'Online pet supply retailer' "Kids' clothing subscription"
'Mom-focused deal site' 'Lifestyle e-commerce'
'UK fashion resale marketplace' 'Crowd-sourced product retailer'
'Gift card marketplace' 'UK fashion rental service' 'UK coupon platform'
'Sample product marketplace' 'Social selling app'
'POS for small retailers' 'Retail rewards app'
'Boutique e-commerce aggregator' 'Shopping tracker app'
'Fashion marketplace app' 'Custom apparel platform'
'Resale clothing subscription' 'Indie used book marketplace'
'Lifestyle rewards program' 'Organic food subscription'
'Fashion deal site' 'Luxury resale marketplace'
'Personal styling service' 'Try-before-you-buy fashion' 'Daily deal site'
'Local deal platform' 'Local service marketplace'
'Experimental retail projects' 'Local shopping platform'
'Niche gadget retailer' 'Indie e-book retailer'
'Personalized asthma/allergy app' 'Doctor search and cost estimation'
'Personalized health coaching' 'Remote medical scribes'
'Telemedicine for hospitals' 'AI-powered telemedicine'
'Health smartwatch' 'Mental health for employees'
'Online therapy (early)' 'Stress monitoring wearable'
'Telemedicine for nursing homes' 'Heart rate analysis app'
'Care coordination platform' 'Medication management app'
'AI eye diagnostics' 'Mobile wallet (health-adjacent)'
'At-home diagnostics' 'Behavioral tech for apps'
'Free telemedicine (early)' 'Cancer trial matching'
'Alzheimer's therapeutic' 'Hearing aids' 'Fitness app'
'Mental health support' 'Precision kidney meds' 'AI for rare diseases'
'Health-based insurance' 'Posture wearable' 'Telemedicine kiosks'
'Subscription telemedicine' 'Fitness app (early)' 'Telemedicine robots'
'Heart monitoring patch' 'Fitness trackers' 'Cancer symptom tracking'
'Doctor matching (early)' 'Digital mental health' 'AI drug repurposing'
'AI skincare diagnostics' 'Care orchestration'
'Blockchain medical records' 'Smart pill dispenser'
'Mental health via data' 'Telehealth birth control'
'AI healthcare automation' 'Diabetes prevention (early)'
'Portable dialysis (early)' 'Digital therapeutics'
'Pre-sorted meds delivery' 'Ingestible sensors' 'Quit smoking app'
'Home vitals device' 'Virtual primary care' 'Healthy meal delivery'
'Digital stethoscope' 'Blood testing tech' 'Tinnitus treatment app'

```
 'Microbiome testing' 'Sleep tracking headband' 'Online personal loans'
 'Micropayments platform' 'Portfolio tracking tool'
 'Crypto payments and stablecoin' 'Personal finance app' 'Mobile wallet'
 'NFT marketplace' 'Crowdfunding for startups' 'Prepaid card payments'
 'Personal crowdfunding' 'Social sentiment for trading'
 'Loans for subprime borrowers' 'P2P lending platform'
 'Coding bootcamp loans' 'Commission-free brokerage'
 'Real estate lending platform' 'Financial data aggregator'
 'Mobile payments platform' 'Online pawn lending' 'Biometric payments'
 'Digital estate planning' 'Payment gateway' 'Debt management app'
 'Prepaid debit card' 'Small biz banking'
 'Personal finance for gig workers' 'Robo-advisor'
 'Neobank with budgeting' 'Financial planning tools'
 'Social finance platform' 'P2P payments (early)' 'Supply chain finance'
 'Social crowdfunding' 'Crowdfunded trusts' 'AI-driven lending'
 'Income-share agreements' 'Group payments feature' 'Student microloans'
 'High-yield savings' 'Payment processing' 'International transfers'
 'Accounts receivable automation' 'Financial data platform'
 'AI underwriting loans']
Unique values in 'How Much They Raised': ['$70M' '$200M' '$40M' '$39M' '$73M'
'$5M' '$850M' '$400M' '$10M' '$2.7B'
 '$30M' '$330M' '$3.5B' '$1.4B' '$930M' '$120M' '$15M+$34M pre-orders'
 '$10M+$403M acquisition' '$3M' '$50M' '$160M' '$150M' '$15M' '$135M'
 '$375M' '$1.5M' '$20M' '$12.8M' '$0.5M' '$9M' '$1.8M' '$300M' '$42.5M'
 '$0.75M' '$41M' '$1M' '$690M' '$45M' '$4.2M' '$1.7B (Dropbox)' '$290M'
 '$3.3M' '$24M' '$104M' '$14M' '$162M (Foursquare)' '$48.5M'
 '$0 (Yahoo $3.6B)' '$104.6M' '$28.5M' '$10.4M' '$4.9M' '$11.5M' '$249.9M'
 '$89.2M' '$680M (Hulu)' '$62.5M' '$48M' '$57.5M (Instagram)' '$240.8M'
 '$350M' '$4.4M' '$120.5M' '$182.3M' '$5.3M' '$277M' '$86M' '$16.1M'
 '$343M (Notion)' '$5.1M' '$59M' '$1.5M (Twitter $645M)'
 '$1.5B (Pinterest)' '$12.5M' '$60M' '$14.5M' '$79M' '$165M' '$56.8M'
 '$12.2M' '$13M' '$35M' '$64M' '$6.7M' '$543M (SoundCloud)' '$2.5M' '$28M'
 '$82.4M (Substack)' '$367M' '$50M (Tinder)' '$26M' '$130.7M'
 '$10M (Trello)' '$125M' '$24.6M' '$147.5M (TuneIn)' '$80.9M (SendGrid)'
 '$645M (Twitter)' '$163M (Udacity)' '$2M (Rdio $125M)' '$61M' '$12M'
 '$8M' '$84M' '$2M' '$41.8M' '$1B+ (Yahoo)' '$142M' '$73.5M' '$25M'
 '$1.4M (Zapier)' '$85.5M (Zendesk)' '$47.2M' '$33M (Zscaler)' '$138M'
 '$171M (Zumper)' '$866M' '$23.5M' '$49M' '$7.3M' '$4.05M' '$58M'
 '$0.525M' '$3.5M' '$100M' '$46.5M' '$1.2M' '$2.2M' '$1.4M' '$95M' '$326M'
 '$5.8M' '$1.8B' '$4M' '$7M' '$6M' '$336M' '$80M' '$17M' '$250M (Gilt)'
 '$33M' '$950M (Groupon)' '$85M' '$76M' '$110M' '$22M' '$225M' '$75M'
 '$70M (Poshmark)' '$500M (RtR)' '$300M (RetailMeNot)' '$65M' '$55M'
 '$300M (ThredUp)' '$150M (Thrive)' '$122M' '$500M (Yelp)'
 '$1.5B (Zappos)' '$32M' '$635M' '$4.5M' '$31M' '$34M' '$161M' '$404M'
 '$lowM' '$316M' '$220M' '$209M' '$136M' '$43M' '$88M' '$78M' '$153M'
 '$21M' '$92M' '$167M' '$115M' '$856M' '$550M' '$409M' '$118M' '$500M'
 '$77M' '$56M' '$105M' '$655M' '$11M' '$0 (Coinbase-funded)' '$10M (est.)'
 '$1B' '$0.4M' '$3M (est.)' '$185M' '$300M (est.)' '$2M (est.)'
```

```
 '$1M (est.)' '$20M (est.)' '$4M (est.)' '$0 (SoFi-funded)'
 '$0 (Square-funded)' '$67M' '$0.5M (est.)' '$144M' '$0 (Venmo-funded)'
 '$5M (est.)' '$0 (Wealthfront-funded)' '$689M' '$112M']
Unique values in 'Why They Failed': ['Lost to DJI and high costs'
 'High costs and competition from Lego/Sphero'
 'Lost to Tesla and quirky design' 'Small market and lost to DJI/Skydio'
 'Lost to Ring/Nest and acquired' 'Lost to Raspberry Pi and small niche'
 'Lost to Tesla and high infra costs'
 'Lost to GE/Philips and slow adoption'
 'Lost to Lego/Sphero and niche focus' 'Lost to Tesla/Nio and high costs'
 'Lost to Toyota/Honda and delays' 'Poor sales vs Apple/Samsung'
 'Mismanagement and Tesla/Rivian competition'
 'Lost to Tesla and supplier failure' 'Lost to DJI and recall issues'
 'Lost to Fitbit/Apple Watch' 'Competition and quality issues'
 'Overkill product vs Ninja/Breville' 'Smartphones caught up'
 'Couldn’t deliver vs DJI' 'Smartphones matched it vs Canon/Nikon'
 'Lost to Ultimaker/Prusa post-acquisition' 'Niche market vs Hach'
 'Lost to Tesla/Ford built-ins' 'Lost to Apple Watch and sold'
 'Lost to Velodyne/Luminar and delays' 'Lost to Universal Robots/Fanuc'
 'Mismanagement and no product vs Nolan/Shoei'
 'Lost to Leap Motion/Oculus and sold'
 'High costs vs Domino’s/Papa John’s'
 'Shut down in 2012 after chaotic sprint; too many games; no focus; cash ran
out'
 "Faded by 2007; lost to Google's algorithm and ad model; sold off"
 'Shut down in 2019; lost to Facebook; AOL mismanagement; sold cheap'
 "Closed in 2012 but pivoted to Instagram; too cluttered; users didn't care"
 'Shut down in 2017; lost to Google Docs; Dropbox Paper; no differentiation'
 'Startup phase "shut down" 2022; monetization alienated users; growth slowed'
 'Closed in 2016; niche; no real problem solved; adoption lagged'
 'Wound down 2023; lost to AWS; Google BigQuery; went private'
 'Shut down 2022; lost to VPNs; Chrome; trust waned'
 'Acquired 2021; "closed" 2022; free tiers didn\'t convert; lost to Coursera'
 'Shut down 2018; too niche; lost to GitHub; LinkedIn'
 'Closed 2014; confused users; lost to Instagram; Apple absorbed'
 "Shut down 2016; lost to Facebook's own features; privacy issues"
 "Closed 2021; couldn't monetize; lost to TikTok"
 'Scaled back 2023; lost to Tableau; Power BI; high costs'
 'Faded 2018; redesign tanked; lost to Reddit; sold cheap'
 "Closed 2019; open-source didn't monetize; lost to React Native"
 'Shut down 2021; lost to Google Docs; Notion'
 'Declined 2019; bloated features; lost to OneNote'
 'Shut down 2015; thin margins; lost to TaskRabbit'
 'Shut down 2023; lost to Getty; Unsplash'
 'Closed 2022; merged into Foursquare; lost to Google Maps'
 'Shut down 2016; bullying scared users; lost to Facebook'
 'Ended 2021; check-ins faded; lost to Instagram'
 'Faded 2014; slow servers; lost to Facebook'
```

```
'Shut down 2005; Yahoo neglect; lost to blogs'
'Closed 2016; Twitter; streaming outpaced; sold off'
'Ended 2018; SAP acquisition; lost to Google login'
"Sold to Spotify 2021; couldn't scale profitably"
'Shut down 2012; lost to Foursquare; Facebook acquired'
'Shut down 2017; lawsuits over piracy; lost to Spotify'
'Acquired 2011; faded 2015; stagnated under Skype'
'Scaled back 2023; lost to Sprout Social; native tools'
'Shut down 2021; Epic acquired; lost to Zoom'
'Faded 2018; lost to Slack; Teams; sold off'
'Shut down 2019; sold to Nippon TV; lost to Netflix'
"Scaled back 2022; free users didn't pay; lost to Zapier"
'Closed 2020; lost to Kindle; Google Books'
'Phased out 2021; lost to TikTok; Meta shifted'
'Startup phase "shut down" 2023; lost to Zendesk'
'Scaled back 2023; lost to Figma; Adobe'
'Shut down 2017; lost to Fitbit; buggy tech'
"Closed 2016; Google bought; couldn't scale solo"
'Shut down 2022; privacy scandals; lost to WhatsApp'
'Sold 2020; overhyped AI; lost to Pearson'
'Shut down 2015; Dropbox bought; stagnated'
"Closed 2019; print declined; events didn't cover costs"
'Shut down 2020; lost to AWS IoT; low visibility'
'Closed 2014; Google bought; Oink flopped'
'Startup phase "shut down" 2023; lost to Amplitude'
'Shut down 2017; lost to Instagram'
"Merged into TikTok 2018; couldn't scale solo"
'Faded 2018; lost to Facebook; News Corp mismanaged'
'Shut down 2003; lost to Microsoft IE'
'Closed 2012; no journalists or readers; misread need'
"Phased out 2023; didn't stand out"
'Shut down 2019; Scroll acquired; lost to Twitter'
'Closed 2018; niche; privacy scandal'
'Shut down 2020; lost to Instagram Live'
'Faded 2018; lost to Instagram; paywall backlash'
'Phased out 2020; absorbed by global Pinterest'
'Sold 2017; lost to AWS; OpenStack free'
'Startup phase "shut down" 2023; lost to Netflix'
'Acquired 2017; "closed" 2018; lost to Evernote'
'Sold to LinkedIn 2013; faded 2017; lost to Flipboard'
'Shut down 2019; too niche; lost to LinkedIn'
'Merged 2020; lost to Google Trends'
'Shut down 2017; lost to Apple/Google search'
"Pivoted 2020; couldn't monetize; lost to Spotify"
"Shut down 2015; lost to Pocket's free model"
'Acquired 2015; "closed" 2016; lost to LinkedIn'
"Shut down 2019; couldn't monetize; lost to MongoDB"
'Acquired 2018; "closed" 2019; lost to Gboard'
```

'Shut down 2014; lost to Chrome; Yahoo bought'
'Shut down 2018; toxicity scared users; lost to Whisper'
'Faded 2020; lost to native sharing'
'Acquired 2014; "closed" 2017; lost to Spotify'
"Phased out 2022; free users didn't convert"
'Shut down 2016; lost to Spotify' 'Closed 2021; lost to Slack bots'
'Shut down 2023; costly; low readership' 'Faded 2020; lost to Zoom'
"Shut down 2014; didn't catch on" 'Faded 2018; lost to Zoom'
'Shut down 2021; confused users'
'Acquired 2012; "closed" 2018; lost to Zoom'
'Startup phase "shut down" 2022; lost to Google Classroom'
'Shut down 2018; lost to Facebook' 'Phased out 2022; free sufficed'
'Faded 2020; Yahoo mismanaged; porn ban'
'Sold 2022; lost to free resources'
"Phased out 2021; free users didn't pay"
'Startup phase "shut down" 2023; lost to Mailchimp'
'Shut down 2020; Google bought parts' 'Shut down 2015; lost to Spotify'
'Shut down 2022; lost to Upwork' 'Shut down 2015; lost to Netflix'
'Shut down 2017; Twitter neglected; lost to Snapchat'
'Faded 1984; lost to Lotus 1-2-3'
'Faded 2020; toxicity; lost to Snapchat' 'Shut down 2019; lost to X'
'Acquired 2020; "closed" as standalone; lost to Signal'
'Faded 2019; lost to Instagram Stories' 'Closed 2018; lost to Tinder'
'Startup phase "shut down" 2022; lost to Lever'
'Shut down 2014; lost to browser sync'
'Acquired 2013; "closed" 2013; lost to Gmail'
'Shut down 2020; spam; lost to Quora' 'Closed 2012; lost to Digg'
'Shut down 2013; Yahoo neglect' 'Shut down 2010; low adoption'
'Acquired 2012; "closed" 2018; lost to Slack'
'Shut down 2018; toxicity; lost to Snapchat'
'Faded 2018; novelty; no depth' 'Faded 2020; lost to Wix'
'Faded 2022; lost to Twitch' 'Phased out 2023; free sufficed'
'Acquired 2012; "closed" 2017; lost to Brightcove'
'Phased out 2023; lost to Intercom' 'Faded 2019; lost to Dropbox'
'Shut down 2020; lost to Supercell' 'Sold 2021; lost to Tableau'
'Acquired 2014; "closed" 2016; lost to Zendesk'
'Phased out 2023; lost to CrowdStrike' 'Sold 2019; lost to Amazon'
'Shut down 2022; robots flopped' 'Phased out 2021; free listings won'
'Faded 2013; lost to Figma' 'Shut down 2017; lost to Epic!'
'Faded 2019; mobile shift; sold 2022'
'Shut down 2008; friction; lost to PayPal'
'Faded 2018; lost to Instagram Stories' 'Faded 2021; lost to LinkedIn'
'Shut down 2017; lost to LinkedIn'
'Acquired 2014; "closed" 2014; lost to Apple'
'Shut down 2012; lost to AWS' 'Sold 2021; lost to Netflix'
'Acquired 2016; "closed" 2020; lost to Discord'
'Sold 2010; faded 2019; lost to Fitbit' 'Faded 2020; lost to Stitch Fix'
'Sold 2022; lost to Veeam' 'Sold 2014; lost to Concur'

'Sold 2019; shut down; lost to Google Maps' 'Closed 2017; lost to Medium'
'Acquired 2015; "closed" 2016; lost to Netlify'
'Sold 2021; privacy regs; lost to Google' 'Closed 2018; lost to Upwork'
'Shut down 2019; lost to Google Photos' 'Closed 2013; lost to X'
'Shut down 2016; lost to Instagram' 'Closed 2016; lost to Candy Crush'
'Sold 2021; lost to Powtoon' 'Sold 2019; lost to Epic!'
'Acquired 2011; shut down; lost to Foursquare'
'Shut down 2013; low retention; funding fell through'
'Closed 2017; high marketing costs; lost to Amazon'
'Closed 2016; financial losses; lost to Kindle'
'Shut down 2018; high costs; lost to eBay; valuation scandal'
"Closed 2017; couldn't scale; lost to Amazon"
'Shut down 2017; high shipping costs; cash burned out'
'Bankruptcy 2021; lost to Amazon; high shipping costs'
'Faded 2022; low adoption; lost to data brokers'
'Closed 2015; dealers resisted; cash ran dry'
'Shut down 2018; low demand; lost to Stitch Fix'
'Closed 2017; lost to Amazon; shipping delays'
'Bankruptcy 2020; overexpansion; quality issues'
'Faded 2023; low-quality goods; lost to Amazon'
'Shut down 2018; soft demand; high logistics costs'
'Closed 2016; local limits; lost to Instacart'
'Shut down 2018; overexpansion; lost to Wayfair'
'Sold 2021; faded 2023; Uber struggled with regs'
'Closed 2019; lost to Wine.com; low margins'
'Shut down 2013; mismanagement; lost to Amazon'
'Closed 2020; niche demand; lost to TVs'
'Sold 2018; overexpansion; lost to Amazon'
'Faded 2019; low conversion; lost to Amazon'
'Closed 2014; low engagement; lost to Instagram'
'Faded 2018; lost to Google/Facebook ads'
'Closed 2017; low adoption; lost to DoorDash'
'Shut down 2016; high costs; sold off'
'Sold 2019; shipping costs; lost to DoorDash'
'Scaled back 2022; high costs; lost to Amazon Fresh'
'Sold 2019; faded 2020; lost to Amazon'
'Phased out 2015; low need; lost to Yelp'
'Startup phase "shut down" 2023; lost to Wayfair'
'Closed 2023; lost to Sephora; high costs'
'Faded 2022; too niche; lost to eBay'
'Shut down 2016; legal issues; lost to Handy'
'Closed 2020; lost to Casper; low scale'
'Faded 2022; lost to Rakuten; low retailer uptake'
'Closed 2015; lost to Amazon; low adoption'
'Closed 2015; lost to Amazon; niche limits'
'Faded 2023; lost to Sephora; saturated market'
'Closed 2018; lost to Rent the Runway; low demand'
'Faded 2022; billing complaints; lost to Nordstrom'

```
'Closed 2018; lost to Groupon; low scale'
'Closed 2002; free delivery killed margins; dot-com bust'
'Faded 2018; lost to Wine.com; low margins'
'Closed 2018; lost to local rivals; high costs'
'Faded 2022; lost to Amazon; niche limits'
'Faded 2022; lost to Amazon; high costs'
'Faded 2021; lost to Sephora; saturated market'
'Sold 2017; faded 2020; Walmart diluted it'
'Closed 2017; logistics costs; lost to eBay'
'Bankruptcy 2017; mismanagement; lost to ASOS'
'Sold 2019; lost to Wayfair; high costs'
'Closed 2020; lost to Amazon; low adoption'
'Faded 2022; lost to Lululemon; overexpansion'
'Faded 2021; low uptake; lost to eBay'
'Faded 2020; lost to Amazon; niche limits'
'Shut down 2000; high shipping costs; dot-com bust'
'Closed 2020; lost to Amazon; low demand'
'Faded 2017; lost to Groupon; deal fatigue'
'Faded 2021; lost to Amazon; low scale'
'Faded 2020; lost to Depop; low UK traction'
'Bankruptcy 2019; high costs; lost to Amazon'
'Faded 2022; lost to Amazon; low margins'
'Faded 2021; lost to local rivals; low demand'
'Faded 2020; lost to local sites; deal fatigue'
'Closed 2017; lost to Amazon; low scale'
'Closed 2016; lost to eBay; low uptake'
'Faded 2020; lost to Square; low scale'
'Faded 2020; sold to Trax; lost to Amazon'
'Faded 2020; lost to Etsy; low scale'
'Faded 2021; lost to Amazon; low adoption'
'Faded 2019; lost to Amazon; high costs'
'Faded 2021; lost to Amazon; low margins'
'Faded 2023; lost to ThredUp core; low demand'
'Faded 2020; lost to Amazon; low scale'
'Faded 2018; lost to Amazon; low uptake'
'Faded 2023; lost to Whole Foods; high costs'
'Closed 2015; lost to Gilt; deal fatigue'
'Faded 2021; lost to The RealReal; low scale'
'Shut down 2019; Nordstrom flopped; lost to Stitch Fix'
'Faded 2022; lost to Amazon Wardrobe; low adoption'
'Shut down 1999; overspent on warehouses; dot-com bust'
'Closed 2015; lost to Amazon; low demand'
'Sold 2010; faded 2016; lost to Amazon; deal fatigue'
'Faded 2019; lost to Groupon; deal fatigue'
'Faded 2017; lost to TaskRabbit; low scale'
'Faded 2018; lost to Amazon core; low impact'
'Faded 2020; lost to Amazon; low uptake'
'Closed 2018; lost to Amazon; low scale'
```

'Faded 2019; lost to Amazon; low demand'
'Small user base and cash shortage'
'Lost to Zocdoc/GoodRx and slow adoption' 'High costs and low demand'
'Lost to software rivals and acquired'
'Outpaced by bigger rivals and acquired' 'Overexpansion and losses'
'Overheating and competition' 'Crowded market and no differentiation'
'Trust issues and competition' 'Lost to bigger players and acquired'
'Slow growth and high costs' 'Monetization and differentiation woes'
'No payer contracts and revenue' 'Lost to free tools and sold'
'Lost to IDx and regulatory delays' 'Leadership and no product'
'Post-COVID crash and costs' 'Lost focus and faded'
'Free model unsustainable' 'Bad pricing and funding'
'Small market and funding' 'Fraud allegations and competition'
'Lost steam and acquired' 'High costs and merged'
'Funding dried up and slow progress' 'Funding crunch and slow results'
'Risk models and scrutiny' 'Lost to Fitbit and costs'
'Lost to software rivals' 'Low retention and pivot'
'Slow retention and pivot' 'Scalability issues and sold'
'Reimbursement woes early' 'Slow adoption and acquired'
'Lack of traction and pivot' 'Slow adoption and misjudgment'
'Slow results and pivot' 'Crowded market and cash out'
'Slow sales and wind-down' 'Lost to Epic and crypto crash'
'High costs and low uptake' 'No actionable insights'
'Layoffs and valuation cuts' 'Reg hurdles and competition'
'Downturn and mismanagement' 'Slow payer adoption and pivot'
'Production snags and competition' 'Reimbursement woes'
'Profitability woes and sold' 'Failed partnership and costs'
'Low retention and adoption' 'Regulatory delays and competition'
'Slow uptake and sold' 'High costs and competition'
'Distribution and funding' 'Fraud and tech didn’t work'
'Limited proof and market' 'Fraudulent billing'
'Competition and no viable model' 'Lost to LendingClub and high defaults'
'Lost to PayPal and low adoption' 'Lost to Mint and sold to TradeKing'
'Lost to Coinbase and market shifts'
'Lost to Mint/Acorns and sold to Goldman'
'No product vs Apple Pay/PayPal' 'Lost to OpenSea and NFT crash'
'Lost to Kickstarter and low ROI' 'Lost to Green Dot and small scale'
'Lost to GoFundMe and niche focus' 'Lost to Bloomberg and low uptake'
'Regulatory fines and OppLoans competition'
'Lost to banks and trust issues' 'Lost to Affirm and sold'
'Lost to Robinhood and closed' 'Lost to Fundrise and slow growth'
'Lost to Bloomberg and early market' 'Lost to Square and mismanagement'
'Lost to traditional pawnshops and niche'
'Lost to PayPal and tech issues' 'Lost to LegalZoom and low uptake'
'Lost to Stripe and sold' 'Lost to Credit Karma and sold'
'Lost to Chime and trust issues' 'Lost to Brex and closed'
'Lost to Even and niche' 'Lost to Wealthfront and sold'
'Lost to Chime/Ally and shut post-BBVA sale'

```
 'Lost to NerdWallet and slow growth' 'Lost to Robinhood and niche'
 'Lost to Venmo and pivoted' 'Lost to Coupa and sold'
 'Lost to GoFundMe and sold' 'Lost to Mint and low scale'
 'Lost to traditional trusts and niche' 'Lost to banks and high defaults'
 'Lost to Lambda School and regs' 'Lost to Splitwise and niche'
 'Lost to SoFi and low scale' 'Lost to Ally and pivoted'
 'Lost to Mint and early market' 'Lost to PayPal and high costs'
 'Lost to Bill.com and sold' 'Lost to Bloomberg and sold'
 'Lost to Upstart and regs']
Unique values in 'Takeaway': ['Drones need simplicity' 'Consumer hardware needs
mass pricing'
 'EVs need mainstream appeal' 'Hardware niches need big adopters'
 'Smart home needs ecosystem power' 'Open-source needs community'
 'Infra needs buy-in' 'Medical hardware needs fast inroads'
 'Edtech needs scale' 'EVs are a giant’s game' 'Cheap cars need delivery'
 'Smartphone giants are unbeatable' 'EV hype needs execution'
 'Luxury EVs need reliability' 'Drones need perfection'
 'Sleep tech must be wearable' 'Fitness hardware needs quality'
 'Hardware must solve real problems' 'Cameras can’t outrun trends'
 'Don’t overpromise hardware' 'Cameras can’t beat smartphones'
 '3D printing needs precision' 'IoT needs a big niche'
 'Auto tech must beat OEMs' 'Smartwatches need ecosystems'
 'LiDAR needs booming market' 'Robotics demands ROI' 'AR needs execution'
 'Wearables need a killer app' 'Food tech needs efficiency'
 'Focus beats frenzy' "Innovation isn't enough"
 'Network effects can crush' 'Pivots can save' 'Stand out or drown'
 'Mission drift costs' "Cool doesn't equal useful" 'Giants eat niches'
 'Trust is currency' "Freemium's a gamble" 'Networks need mass'
 "Hype can't save fit" "Don't bet on turf" 'Timing and scale brutal'
 'Big bets need wins' 'User loyalty fragile' "Free doesn't pay"
 'Stick to strengths' 'Simplicity beats bloat' 'Margins matter'
 'Scale beats niche' "Data's only gold if sold" "Toxicity's a sword"
 'Trends fade fast' 'Speed and UX trump' 'Adapt or atrophy'
 'Niche trends risky' 'Regs and giants choke' "Content's costly"
 'Winners take all' "Legality's key" 'Acquisition dead end'
 "Cash can't outrun" "Peaks don't last" 'Ecosystems win'
 'Local beats generic' 'Free can trap' 'Niche needs demand'
 'Features can fade' 'Giants catch up' 'Better beats big'
 'Execution outranks cash' 'Small can sell' 'Trust is king'
 'Prove the pudding' 'Acquisition stalls' "Passion doesn't pay"
 'Giants dominate IoT' 'Validate before build' 'Lead or lose'
 "Stars don't guarantee" 'Small fish swallowed' 'UX trumps leads'
 'Titans tilt field' 'Markets need sides' 'Features need identity'
 'Platforms evolve' "Privacy's tightrope" 'Trends shift platforms'
 "Don't alienate base" 'Global trumps silos' 'Free kills premium'
 'Niche caps growth' 'Free limits scale' 'Buyers can bury'
 'Niche needs scale' "Differentiation's key" 'Solve a real pain'
 'Monetize or fade' 'Free wins over paid' 'Big fish eat ideas'
```

```
'Open source needs pay' 'Small gets scooped' 'Simple beats quirky'
"Anonymity's messy" 'Platforms steal thunder' 'Small fries eaten'
'Free roots deep' 'Crowds beat cults' 'Giants gobble niches'
"Local's tough sell" "Cash can't crowd out" "Pedigree isn't enough"
'Basics beat quirky' 'Stick to your lane' 'APIs eat pioneers'
'Niche edtech ceilings' 'Social kills local' 'Free trumps upsell'
'Mismanagement burns' 'Free eats paid' 'Free roots stifle'
'Giants crowd out' 'Focus drifts kill' 'Stick to core'
'Extensions need traction' "Timing's everything" 'Neglect kills hits'
'Innovate or die' "Anonymity's gamble" 'Middlemen get cut'
"Privacy's crowded" 'Fads burn out' 'Giants rule dating' 'Scale or stall'
'Tech shifts bury' 'Features swallow' 'Quality beats chaos'
'Latecomers lose' 'Neglect kills legacy' "Timing's critical"
'Buyouts can bury' "Anonymity's beast" "Gimmicks don't last"
'Polish beats early' 'Scale owns live' 'Free caps premium'
'Small exits fast' 'Features fade' 'Free evolves fast'
'Giants crush niches' 'Small sells' 'Sidekicks fade' 'Deals die fast'
'Tech needs results' 'Free trumps tiers' 'Old tools rust'
"Kids' tech brutal" 'Platforms shift' 'Friction kills'
'Trends trump toys' 'Scale beats middlemen' "Platforms don't mix"
'Buyouts can end' 'Giants rewrite rules' 'Niches get nabbed'
'Newcomers steal' 'Tiny bets fade' 'Scale trumps style' 'Old tech rusts'
'Free beats curated' 'Giants own content' 'Small exits quick'
'Regs reshape ads' 'Giants own storage' 'Fads need legs'
"Quirks don't scale" 'Hits beat casual' 'Scale beats tools'
'Giants own fandom' "Kids' ed needs scale" 'Early exits end'
'Curation needs clout' 'Retention is king' "Niche doesn't defend"
'Adapt or die' 'Trust and economics matter' 'Niche retail bleeds'
'Costs can kill' 'Scale trumps convenience' "Tech alone doesn't sell"
'Resistance stalls disruption' 'Niche needs traction'
'Speed beats novelty' 'Quality over growth' "Cheap doesn't last"
'Niche needs a hook' 'Local limits growth' 'Cash burn kills curation'
'Niche needs volume' 'Execution beats ethos' "Hardware's a gamble"
'Growth needs profit' 'Curation needs cash flow'
'Social needs stickiness' 'Giants own ads' 'Supply needs demand'
'Luxury needs volume' 'Logistics eat profits' 'Freshness costs'
'Subscriptions need stickiness' 'Timing trumps deals'
'Scale beats service' "Personalization's pricey" 'Niche needs traffic'
'Service needs trust' 'Niche needs edge' 'Loyalty needs scale'
'Innovation needs buyers' 'Subscriptions saturate' 'Rentals need pull'
'Trust trumps trends' 'Deals need reach' "Free's a trap"
'Niche needs profit' 'Local giants win' 'Beauty saturates'
'Identity matters' 'Logistics sink niches' 'Growth needs control'
'Scale beats decor' 'AI needs buyers' 'Fashion needs edge'
'Rentals need scale' 'Niche caps tech' 'Margins beat hype'
'Niche needs kids' 'Deals fade fast' 'Lifestyle needs reach'
'Resale needs local' "Crowds don't profit" 'Gift cards need scale'
'Rentals need fit' 'Coupons fade too' 'Samples need reach'
```

```
'Social needs buyers' 'POS needs giants' 'Loyalty needs giants'
'Boutiques need reach' 'Tracking needs pull' 'Fashion needs scale'
'Custom needs profit' 'Resale needs core' 'Books need giants'
'Rewards need pull' 'Organic needs scale' 'Deals tire out'
'Luxury needs reach' 'Curation needs scale' 'Try needs buyers'
'Scale needs demand' 'Kids need scale' 'Deals lose steam'
'Deals fade out' 'Services need reach' 'Experiments need wins'
'Local needs pull' 'Gadgets need giants' 'Books need scale'
'Niche apps need big audiences' 'Narrow focus beats broad'
'Premium needs mass market' 'Flexibility beats rigidity'
'Niche needs a moat' 'Quality must match scale'
'Reliability is make-or-break' 'Unique value is critical'
'Trust comes first' 'Standout use case needed' 'Execution must scale'
'Validation is key' 'Need payer buy-in' 'Defensible revenue matters'
'Regulatory wins are key' 'Execution over promises' 'Plan for cycles'
'Focus wins in health' 'Monetization is key'
'Validate willingness to pay' 'Niche needs deep pockets'
'Killer edge needed' 'Profitability matters' 'Timing is critical'
'Deliver or downsize' 'Math and compliance matter' 'Mass appeal needed'
'Sticky engagement needed' 'Evolve or lose' 'Plan for acquisition'
'Payer alignment is key' 'Differentiate or die' 'Traction matters'
'Validate demand early' 'Plan for long cycles' 'Quick wins matter'
'Validate niche early' 'Budget for long runways'
'Real-world buy-in needed' 'Simple beats complex' 'Prove outcomes'
'Growth must match fundamentals' 'Scale and compliance' 'Adapt to shifts'
'Payers drive success' 'Flawless execution needed'
'Therapeutics need economics' 'Scale or get gobbled'
'Don't bet on one partner' 'Stickiness beyond novelty'
'Speed to market matters' 'Big clients fast' 'Need cost advantage'
'Mission needs business model' 'Transparency is key' 'Evidence matters'
'Ethics are non-negotiable' 'Broad appeal needed'
'Lending needs risk balance' 'Micropayments need mass use'
'Finance tools need scale' 'Crypto needs stability'
'Finance apps need edge' 'NFTs need timing' 'Crowdfunding needs hits'
'Prepaid needs reach' 'Personal funding needs appeal'
'Trading tools need traction' 'Fintech can't skirt regs'
'P2P needs trust' 'Small niches get swallowed' 'Brokerages need volume'
'Lending needs scale' 'Timing is key' 'Payments need execution'
'Niche lending needs demand' 'Biometrics need reliability'
'Estate tech needs traction' 'Payments need scale' 'Debt tools need edge'
'Prepaid needs trust' 'Banking needs scale' 'Niche finance needs reach'
'Robo-advisors need edge' 'Neobanks need independence'
'Planning needs traction' 'Social finance needs scale'
'Payments need dominance' 'Finance needs edge' 'Crowdfunding needs mass'
'Finance apps need reach' 'Lending needs stability' 'ISAs need traction'
'Niche payments need use' 'Microloans need reach' 'Savings need edge'
'Timing matters' 'Transfers need edge' 'Automation needs scale'
'Data needs dominance' 'Lending needs compliance']
```

```
Unique values in 'Giants': [1 0]
Unique values in 'No Budget': [1 0]
Unique values in 'Competition': [1 0]
Unique values in 'Poor Market Fit': [0 1]
Unique values in 'Acquisition Stagnation': [0 1]
Unique values in 'Platform Dependency': [0 1]
Unique values in 'Monetization Failure': [0 1]
Unique values in 'Niche Limits': [0 1]
Unique values in 'Execution Flaws': [0 1]
Unique values in 'Trend Shifts': [0 1]
Unique values in 'Toxicity/Trust Issues': [0 1]
Unique values in 'Regulatory Pressure': [0 1]
Unique values in 'Overhype': [0. 1.]
```

[3]:
```
data = data.drop(columns=['Name', 'What They Did', 'Why They Failed',␣
 ↪'Takeaway'])
data['Overhype'] = data['Overhype'].astype(int)
binary_columns = ['Giants', 'No Budget', 'Competition', 'Poor Market Fit',␣
 ↪'Acquisition Stagnation', 'Platform Dependency', 'Monetization Failure',␣
 ↪'Niche Limits', 'Execution Flaws', 'Trend Shifts', 'Toxicity/Trust Issues',␣
 ↪'Regulatory Pressure', 'Overhype']
print(data.head(5))
```

```
          Sector Years of Operation How Much They Raised  Giants  No Budget  \
0  Manufacturing          2011-2018                 $70M       1          1
1  Manufacturing          2010-2019                $200M       1          1
2  Manufacturing          2005-2011                 $40M       1          0
3  Manufacturing          2008-2019                 $39M       1          0
4  Manufacturing          2012-2017                 $73M       1          0

   Competition  Poor Market Fit  Acquisition Stagnation  Platform Dependency  \
0            1                0                       0                    0
1            1                0                       0                    0
2            1                1                       0                    0
3            1                1                       0                    0
4            1                0                       1                    0

   Monetization Failure  Niche Limits  Execution Flaws  Trend Shifts  \
0                     0             0                0             0
1                     0             0                0             0
2                     0             0                0             0
3                     0             1                0             0
4                     0             0                0             0

   Toxicity/Trust Issues  Regulatory Pressure  Overhype
0                      0                    0         0
1                      0                    0         0
2                      0                    0         0
```

```
3                     0                     0         0
4                     0                     0         0
```

[4]:
```
def parse_years(row):
    years_str = str(row)
    # Case 1: Format like "6 (2011-2017)"
    if '(' in years_str and ')' in years_str:
        lifespan = re.search(r'(\d+)\s*\(', years_str)
        founded = re.search(r'\((\d+)-', years_str)
        shutdown = re.search(r'-(\d+)\)', years_str)
        return (int(lifespan.group(1)) if lifespan else np.nan,
                int(founded.group(1)) if founded else np.nan,
                int(shutdown.group(1)) if shutdown else np.nan)
    # Case 2: Format like "2011-2017"
    elif '-' in years_str and '(' not in years_str:
        match = re.search(r'(\d+)-(\d+)', years_str)
        if match:
            founded, shutdown = int(match.group(1)), int(match.group(2))
            lifespan = shutdown - founded
            return lifespan, founded, shutdown
    return np.nan, np.nan, np.nan

# First create temporary columns for years
years_data = data['Years of Operation'].apply(parse_years).apply(pd.Series)
years_data.columns = ['Years of Operation', 'start_year', 'end_year']

# Update the dataframe with the parsed values
data['Years of Operation'] = years_data['Years of Operation']
# data['start_year'] = years_data['start_year']
# data['end_year'] = years_data['end_year']
```

[5]:
```
# Clean Funding: Convert 'How Much They Raised' to numeric values in millions
 ↪($M)
def clean_funding(x):
    if pd.isna(x): return 0
    match = re.search(r'\$(\d*\.?\d+)([MB])', x)
    if match:
        value, unit = float(match.group(1)), match.group(2)
        return value * 1000 if unit == 'B' else value  # Convert billions to
 ↪millions
    return 0
data['Funding ($M)'] = data['How Much They Raised'].apply(clean_funding)
data = data.drop(columns=['How Much They Raised'])

print(data.head(5))
```

```
          Sector  Years of Operation  Giants  No Budget  Competition  \
0  Manufacturing                   7       1          1            1
```

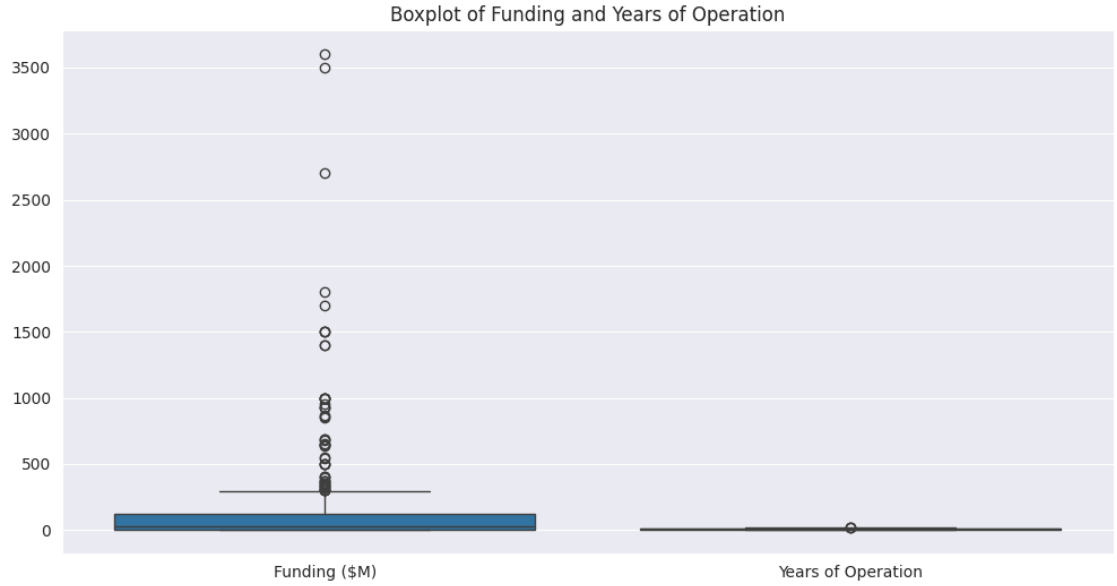
```
1  Manufacturing                   9       1          1            1
2  Manufacturing                   6       1          0            1
3  Manufacturing                  11       1          0            1
4  Manufacturing                   5       1          0            1

   Poor Market Fit  Acquisition Stagnation  Platform Dependency  \
0                0                       0                    0
1                0                       0                    0
2                1                       0                    0
3                1                       0                    0
4                0                       1                    0

   Monetization Failure  Niche Limits  Execution Flaws  Trend Shifts  \
0                     0             0                0             0
1                     0             0                0             0
2                     0             0                0             0
3                     0             1                0             0
4                     0             0                0             0

   Toxicity/Trust Issues  Regulatory Pressure  Overhype  Funding ($M)
0                      0                    0         0          70.0
1                      0                    0         0         200.0
2                      0                    0         0          40.0
3                      0                    0         0          39.0
4                      0                    0         0          73.0
```

0.1 There's no need to perform this on the other numeric columns as they are binary and no distribution transformation is needed.

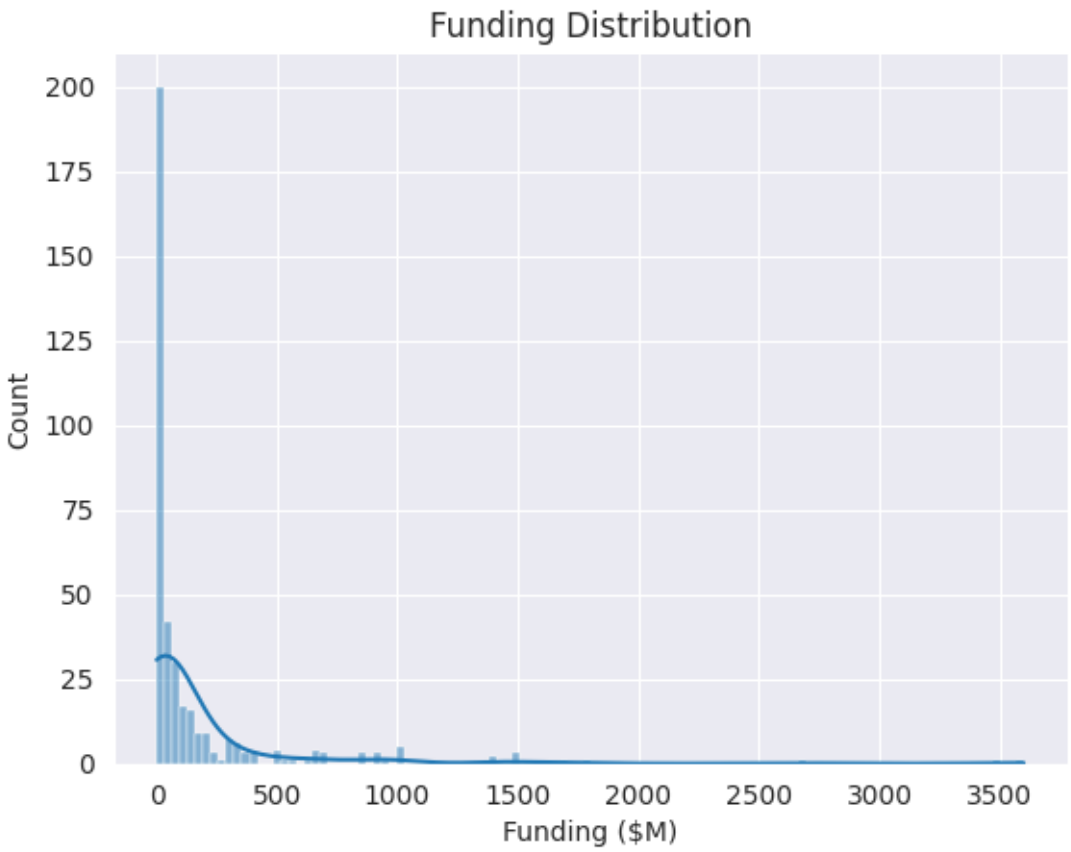
[6]:
```python
# Statistics on Funding and Years of Operation to determine if we need to␣
 ↪normalize or standardize or nothing
print(data[['Funding ($M)', 'Years of Operation']].describe())
# Check for outliers in Funding and Years of Operation
plt.figure(figsize=(12, 6))
sns.boxplot(data=data[['Funding ($M)', 'Years of Operation']])
plt.title('Boxplot of Funding and Years of Operation')
plt.show()
# Check for skewness
print("Skewness of Funding:", data['Funding ($M)'].skew())
print("Skewness of Years of Operation:", data['Years of Operation'].skew())
# Check for normality
sns.histplot(data['Funding ($M)'], kde=True)
plt.title('Funding Distribution')
plt.xlabel('Funding ($M)')
plt.show()
sns.histplot(data['Years of Operation'], kde=True)
plt.title('Years of Operation Distribution')
```

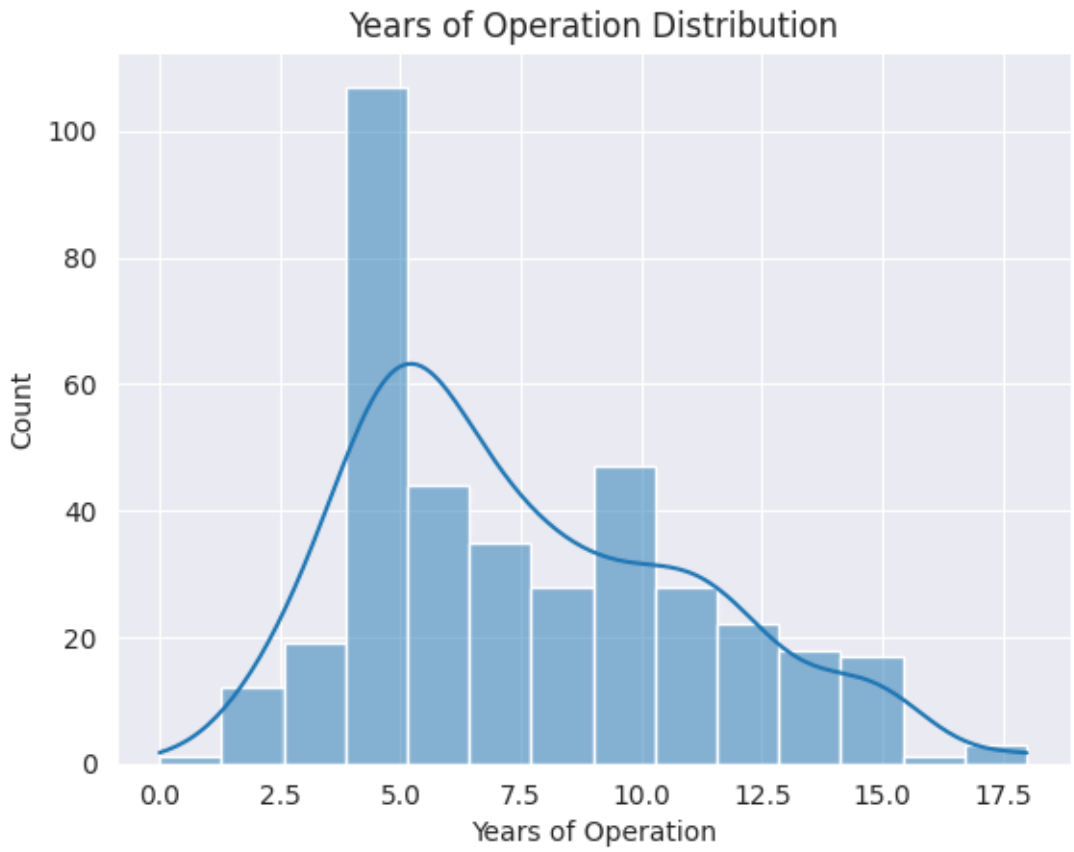
```
plt.xlabel('Years of Operation')
plt.show()
```

```
       Funding ($M)  Years of Operation
count    382.000000          382.000000
mean     161.022971            7.575916
std      394.164114            3.556554
min        0.000000            0.000000
25%        5.000000            5.000000
50%       29.250000            7.000000
75%      121.625000           10.000000
max     3600.000000           18.000000
```

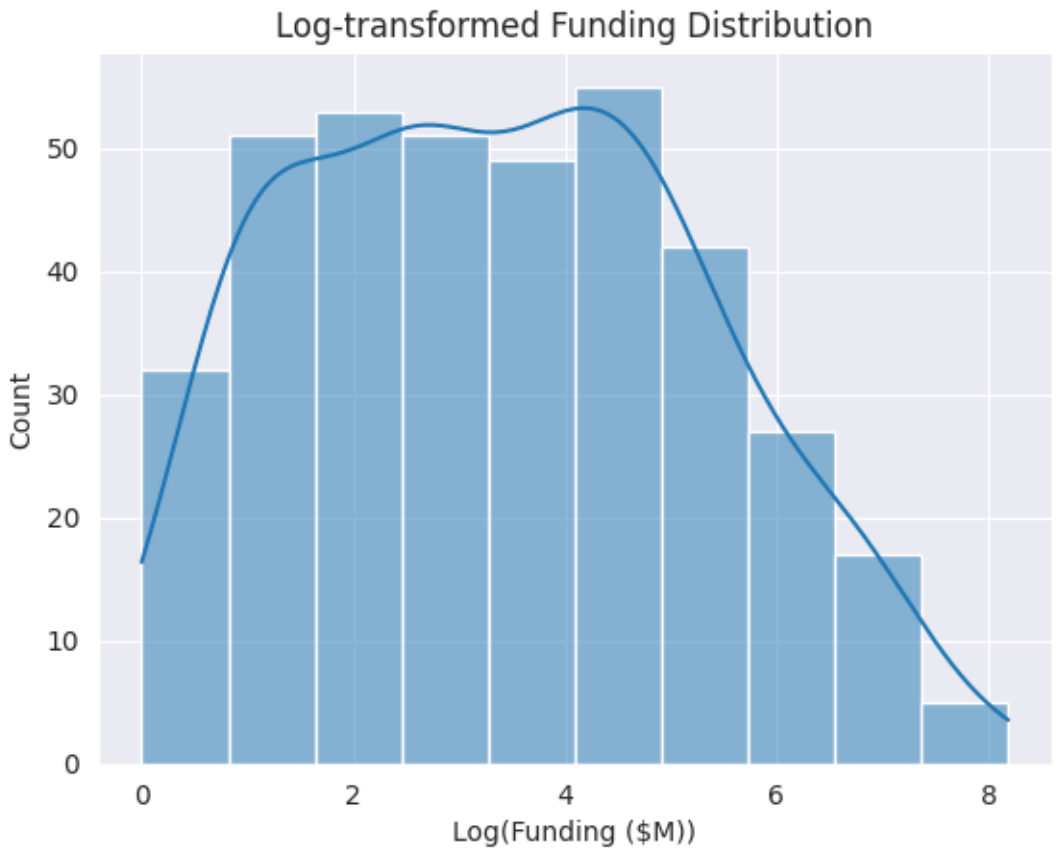



```
Skewness of Funding: 5.159243939703619
Skewness of Years of Operation: 0.6276991129508875
```

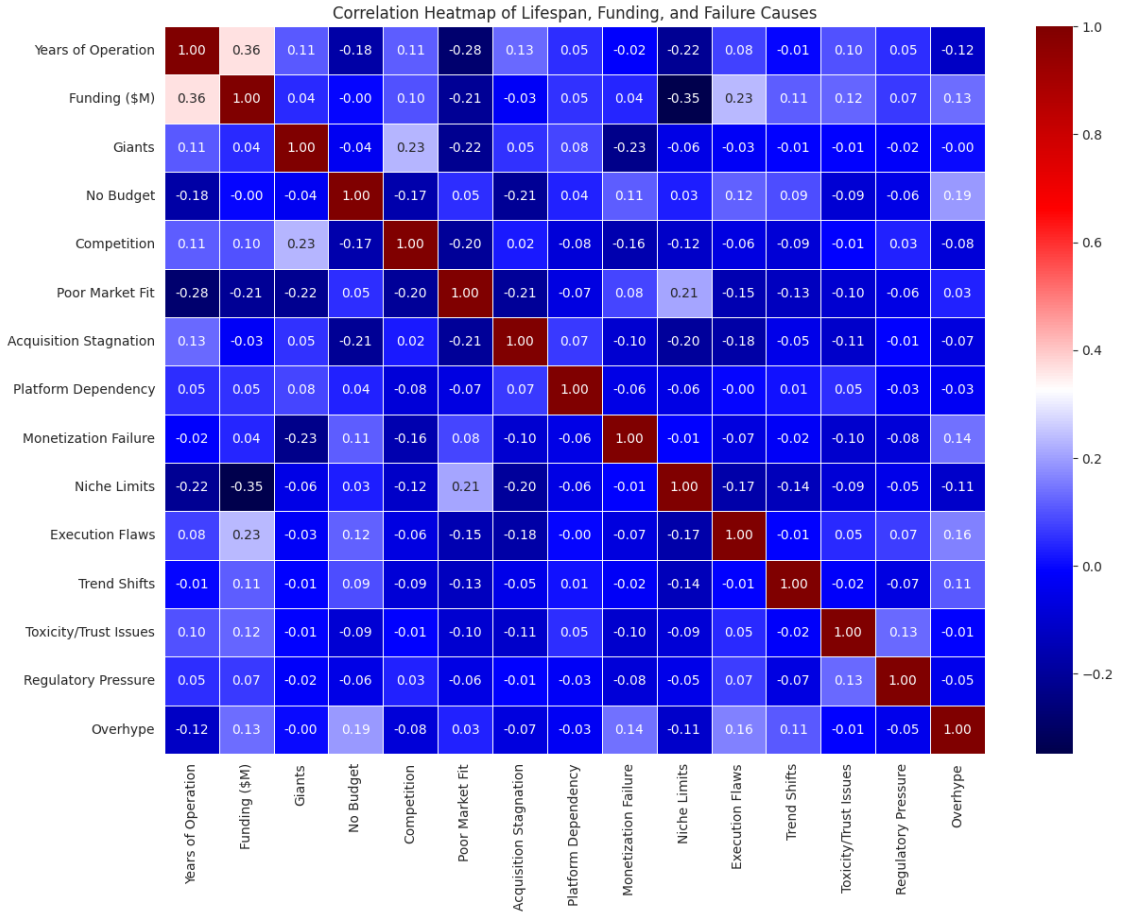

Funding Distribution
Count
200
175
150
125
100
75
50
25
0
0
500
1000
1500
2000
2500
3000
3500
Funding ($M)



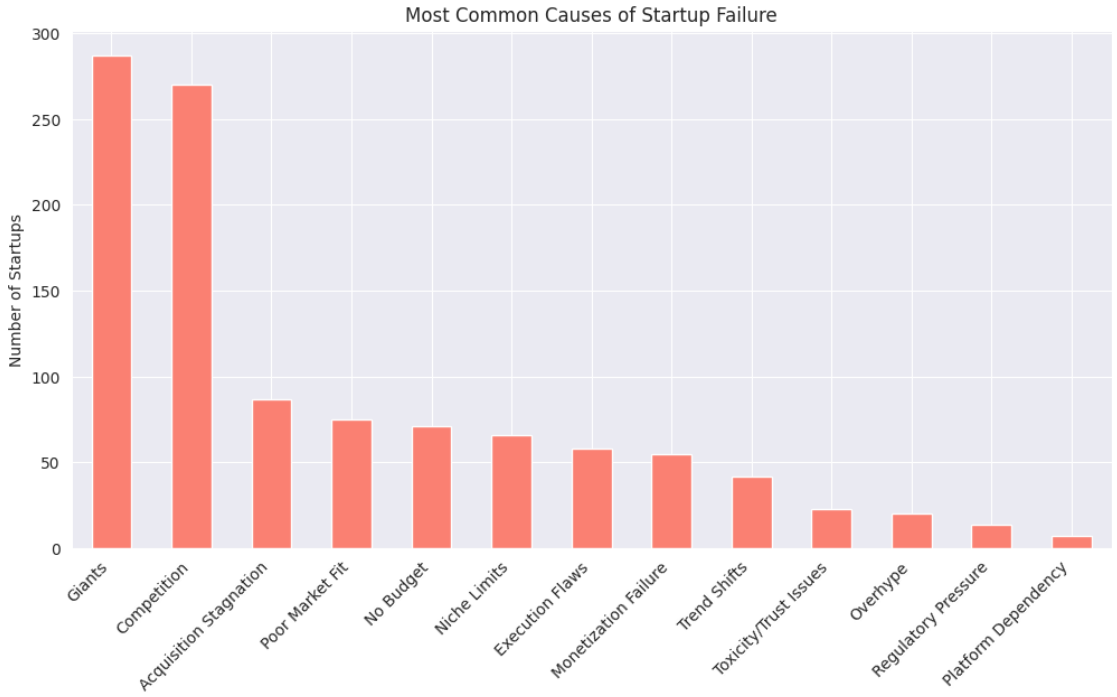
[7]:
```
# Log transform funding, years of operation is fine
data['Funding ($M)'] = np.log1p(data['Funding ($M)'])
# Now show the distribution again
sns.histplot(data['Funding ($M)'], kde=True)
plt.title('Log-transformed Funding Distribution')
plt.xlabel('Log(Funding ($M))')
plt.show()
```



[8]:
```
# Select columns for correlation
corr_cols = ['Years of Operation', 'Funding ($M)'] + binary_columns
corr_matrix = data[corr_cols].corr()
# Generate Heatmap
plt.figure(figsize=(14, 10))
sns.heatmap(corr_matrix, annot=True, cmap='seismic', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap of Lifespan, Funding, and Failure Causes')
plt.show()
```



[9]:
```
failure_sums = data[binary_columns].sum().sort_values(ascending=False)
plt.figure(figsize=(12, 6))
failure_sums.plot(kind='bar', color='salmon')
plt.title('Most Common Causes of Startup Failure')
plt.ylabel('Number of Startups')
plt.xticks(rotation=45, ha='right')
plt.show()
```



[10]:
```
print(data.columns)
# Show unique data types within each column
for col in data.columns:
    unique_types = data[col].apply(type).unique()
    print(f"Unique data types in '{col}': {unique_types}")
```

```
Index(['Sector', 'Years of Operation', 'Giants', 'No Budget', 'Competition',
       'Poor Market Fit', 'Acquisition Stagnation', 'Platform Dependency',
       'Monetization Failure', 'Niche Limits', 'Execution Flaws',
       'Trend Shifts', 'Toxicity/Trust Issues', 'Regulatory Pressure',
       'Overhype', 'Funding ($M)'],
      dtype='object')
Unique data types in 'Sector': [<class 'str'>]
Unique data types in 'Years of Operation': [<class 'int'>]
Unique data types in 'Giants': [<class 'int'>]
Unique data types in 'No Budget': [<class 'int'>]
Unique data types in 'Competition': [<class 'int'>]
Unique data types in 'Poor Market Fit': [<class 'int'>]
Unique data types in 'Acquisition Stagnation': [<class 'int'>]
Unique data types in 'Platform Dependency': [<class 'int'>]
Unique data types in 'Monetization Failure': [<class 'int'>]
Unique data types in 'Niche Limits': [<class 'int'>]
Unique data types in 'Execution Flaws': [<class 'int'>]
Unique data types in 'Trend Shifts': [<class 'int'>]
Unique data types in 'Toxicity/Trust Issues': [<class 'int'>]
```

```
Unique data types in 'Regulatory Pressure': [<class 'int'>]
Unique data types in 'Overhype': [<class 'int'>]
Unique data types in 'Funding ($M)': [<class 'float'>]
```

[11]:
```
one_hot = pd.get_dummies(data['Sector'], prefix='Sector', dtype=int)
data = pd.concat([data, one_hot], axis=1)
data = data.drop(columns=['Sector'])
print(data.columns)
print(data.head(5))
```

```
Index(['Years of Operation', 'Giants', 'No Budget', 'Competition',
       'Poor Market Fit', 'Acquisition Stagnation', 'Platform Dependency',
       'Monetization Failure', 'Niche Limits', 'Execution Flaws',
       'Trend Shifts', 'Toxicity/Trust Issues', 'Regulatory Pressure',
       'Overhype', 'Funding ($M)', 'Sector_Finance and Insurance',
       'Sector_Health Care', 'Sector_Information', 'Sector_Manufacturing',
       'Sector_Retail Trade'],
      dtype='object')
   Years of Operation  Giants  No Budget  Competition  Poor Market Fit  \
0                   7       1          1            1                0
1                   9       1          1            1                0
2                   6       1          0            1                1
3                  11       1          0            1                1
4                   5       1          0            1                0

   Acquisition Stagnation  Platform Dependency  Monetization Failure  \
0                       0                    0                     0
1                       0                    0                     0
2                       0                    0                     0
3                       0                    0                     0
4                       1                    0                     0

   Niche Limits  Execution Flaws  Trend Shifts  Toxicity/Trust Issues  \
0             0                0             0                      0
1             0                0             0                      0
2             0                0             0                      0
3             1                0             0                      0
4             0                0             0                      0

   Regulatory Pressure  Overhype  Funding ($M)  Sector_Finance and Insurance  \
0                    0         0      4.262680                             0
1                    0         0      5.303305                             0
2                    0         0      3.713572                             0
3                    0         0      3.688879                             0
4                    0         0      4.304065                             0

   Sector_Health Care  Sector_Information  Sector_Manufacturing  \
0                   0                   0                     1
```

```
1                    0                    0                    1
2                    0                    0                    1
3                    0                    0                    1
4                    0                    0                    1

   Sector_Retail Trade
0                    0
1                    0
2                    0
3                    0
4                    0
```

3. Use the sample code for the Ensemble classifier and modify it to work with the Startup dataset.

[12]:
```
# print(data.columns)
# output_variable = 'Years of Operation' # Accuracy = 0.2
# output_variable = 'Funding ($M)' # Accuracy = unable, regression problem
# output_variable = 'Overhype' # Accuracy = 0.96
# output_variable = 'Poor Market Fit' # Accuracy = 0.86
# output_variable = 'Execution Flaws' # Accuracy = 0.81
# Going to just plot all of them
# X = data.drop(output_variable, axis=1)
# Y = data[output_variable]
# X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)

results = {}

for output_variable in binary_columns:
    # Skip if this column has already been evaluated
    if output_variable in results:
        continue

    X = data.drop(output_variable, axis=1)
    y = data[output_variable]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

    # Initialize models
    bag_clf = BaggingClassifier(
        DecisionTreeClassifier(random_state=42), n_estimators=500,
        bootstrap=True, n_jobs=-1, random_state=40)

    dt_clf = DecisionTreeClassifier(random_state=42)

    rf_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16,␣
 ↪n_jobs=-1, random_state=42)
```

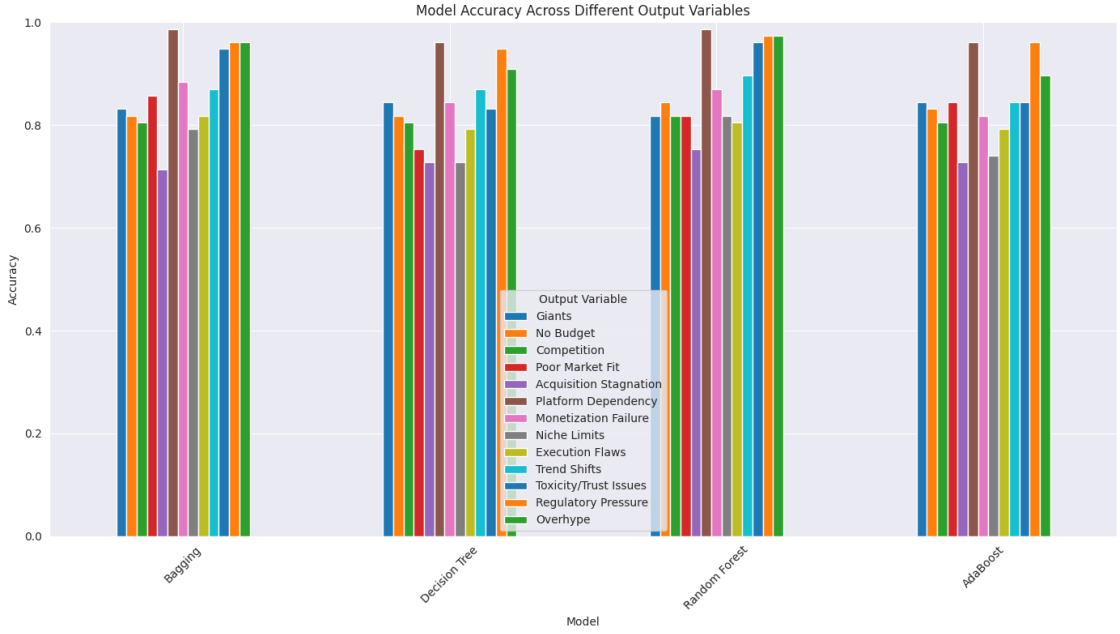
```
    ab_clf = AdaBoostClassifier(
        DecisionTreeClassifier(random_state=42), n_estimators=500,
        learning_rate=1.0, random_state=42)

    # Train and evaluate
    models = {
        'Bagging': bag_clf,
        'Decision Tree': dt_clf,
        'Random Forest': rf_clf,
        'AdaBoost': ab_clf
    }

    output_results = {}
    for name, model in models.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        output_results[name] = accuracy

    results[output_variable] = output_results
    print(f"Completed evaluation for: {output_variable}, Best accuracy:␣
 ↪{max(output_results.values()):.4f}")

# Convert the results to a DataFrame for easier visualization
results_df = pd.DataFrame(results)
```

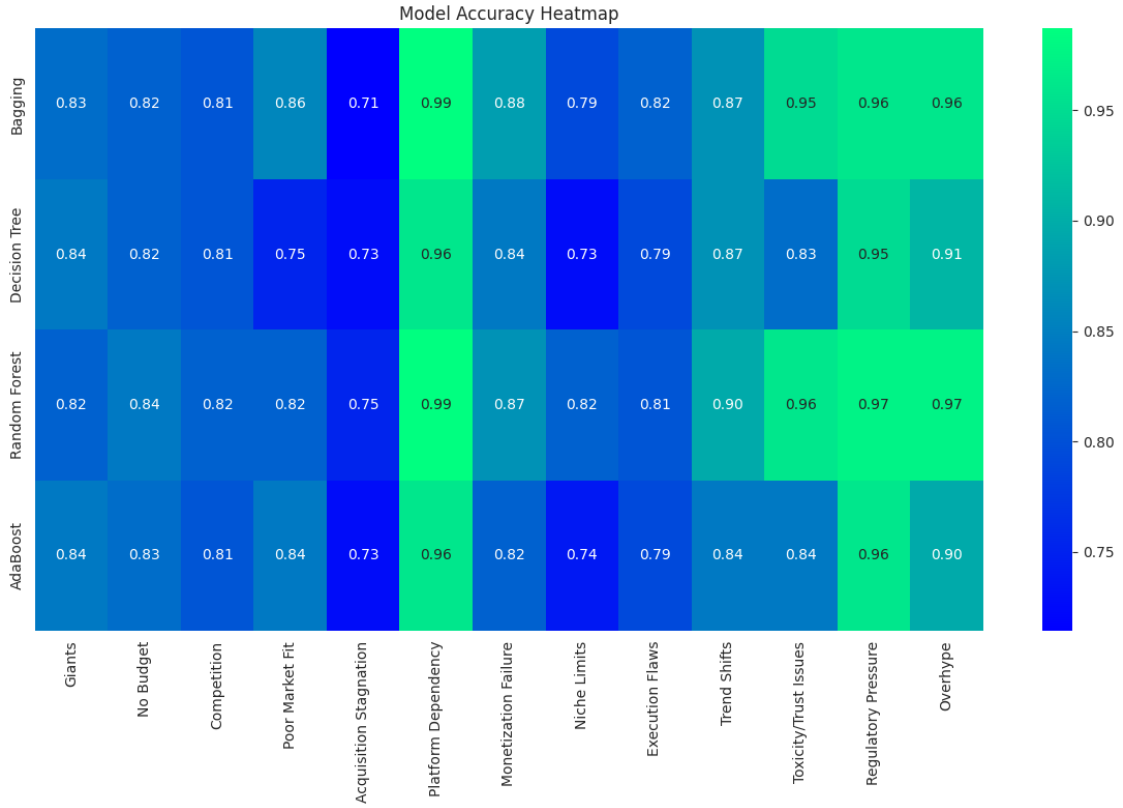
```
Completed evaluation for: Giants, Best accuracy: 0.8442
Completed evaluation for: No Budget, Best accuracy: 0.8442
Completed evaluation for: Competition, Best accuracy: 0.8182
Completed evaluation for: Poor Market Fit, Best accuracy: 0.8571
Completed evaluation for: Acquisition Stagnation, Best accuracy: 0.7532
Completed evaluation for: Platform Dependency, Best accuracy: 0.9870
Completed evaluation for: Monetization Failure, Best accuracy: 0.8831
Completed evaluation for: Niche Limits, Best accuracy: 0.8182
Completed evaluation for: Execution Flaws, Best accuracy: 0.8182
Completed evaluation for: Trend Shifts, Best accuracy: 0.8961
Completed evaluation for: Toxicity/Trust Issues, Best accuracy: 0.9610
Completed evaluation for: Regulatory Pressure, Best accuracy: 0.9740
Completed evaluation for: Overhype, Best accuracy: 0.9740
```

[13]:
```
# Plotting
plt.figure(figsize=(14, 8))
results_df.plot(kind='bar', figsize=(14, 8))
plt.title('Model Accuracy Across Different Output Variables')
plt.xlabel('Model')
plt.ylabel('Accuracy')
```

```
plt.xticks(rotation=45)
plt.ylim(0, 1)
plt.legend(title='Output Variable')
plt.tight_layout()
plt.show()

# Also create a heatmap for better visualization
plt.figure(figsize=(12, 8))
sns.heatmap(results_df, annot=True, cmap='winter', fmt='.2f')
plt.title('Model Accuracy Heatmap')
plt.tight_layout()
plt.show()
```

```
<Figure size 1400x800 with 0 Axes>
```





4. Compare the performance a decision tree, bagging classifier, random forest and a boosting classifier using all default settings and configuration used in the sample code.

0.1.1 Using 'Overhype' as the output variable to create an Overhype predictor. Overhype is a good candidate as it produces accurate classifiers, and has very low counts in the above EDA. By taking in the other features we can accurately predict if a startup is overhyped and therefore should not receive additional funding.

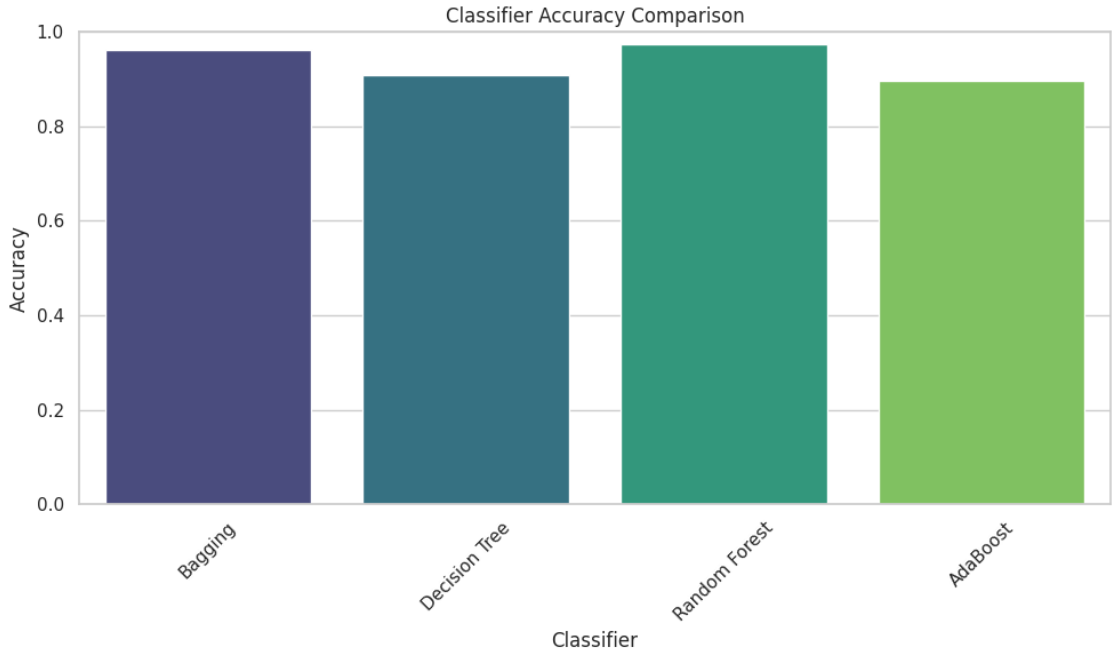
[14]:
```
output_variable = 'Overhype'
X = data.drop(output_variable, axis=1)
Y = data[output_variable]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(random_state=42), n_estimators=500,
    bootstrap=True, n_jobs=-1, random_state=40)
bag_clf.fit(X_train, Y_train)
bag_y_pred = bag_clf.predict(X_test)
```

```
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, Y_train)
dt_y_pred = dt_clf.predict(X_test)

rf_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1,␣
 ↪random_state=42)
rf_clf.fit(X_train, Y_train)
rf_y_pred = rf_clf.predict(X_test)
rf_y_prob = rf_clf.predict_proba(X_test)

ab_clf = AdaBoostClassifier(
    DecisionTreeClassifier(random_state=42), n_estimators=500,
    learning_rate=1.0, random_state=42)
ab_clf.fit(X_train, Y_train)
ab_y_pred = ab_clf.predict(X_test)

models_to_compare_in_step_7 = {
    'Bagging': bag_clf,
    'Decision Tree': dt_clf,
    'Random Forest': rf_clf,
    'AdaBoost': ab_clf
}

# Calculate and plot the accuracy scores
bag_accuracy = accuracy_score(Y_test, bag_y_pred)
dt_accuracy = accuracy_score(Y_test, dt_y_pred)
rf_accuracy = accuracy_score(Y_test, rf_y_pred)
ab_accuracy = accuracy_score(Y_test, ab_y_pred)
print(f"Bagging Classifier Accuracy: {bag_accuracy:.4f}")
print(f"Decision Tree Classifier Accuracy: {dt_accuracy:.4f}")
print(f"Random Forest Classifier Accuracy: {rf_accuracy:.4f}")
print(f"AdaBoost Classifier Accuracy: {ab_accuracy:.4f}")


sns.set(style="whitegrid")
# Data for plotting
methods = ['Bagging', 'Decision Tree', 'Random Forest', 'AdaBoost']
accuracies = [bag_accuracy, dt_accuracy, rf_accuracy, ab_accuracy]

# Create a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=methods, y=accuracies, palette="viridis")
plt.title('Classifier Accuracy Comparison')
plt.xlabel('Classifier')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Bagging Classifier Accuracy: 0.9610
Decision Tree Classifier Accuracy: 0.9091
Random Forest Classifier Accuracy: 0.9740
AdaBoost Classifier Accuracy: 0.8961
```

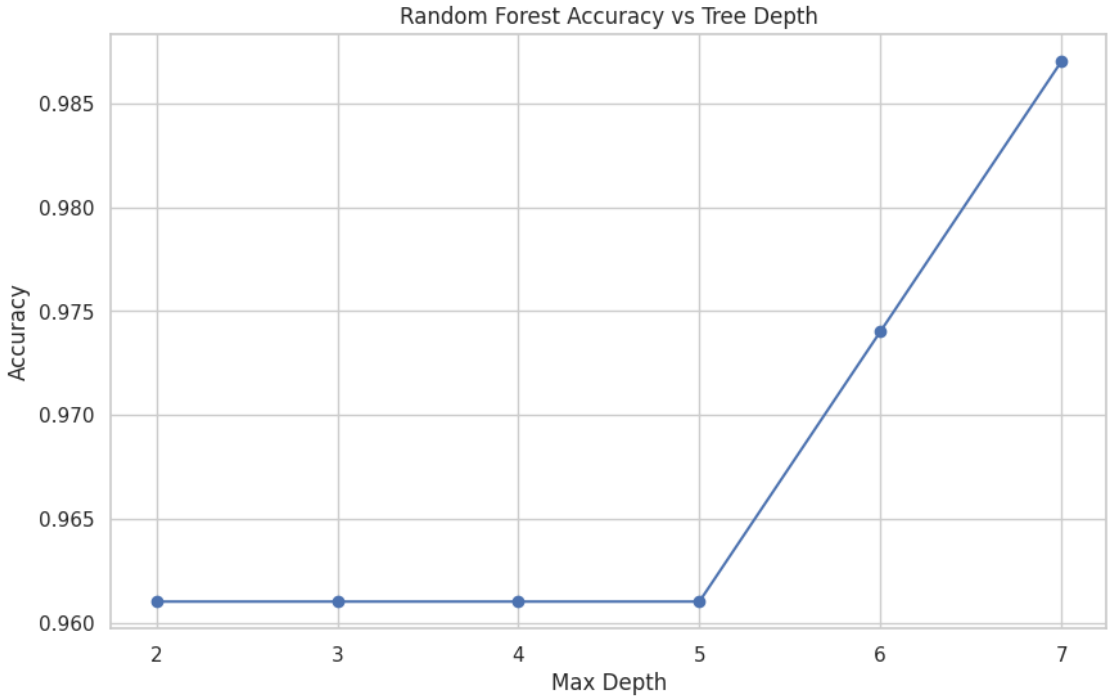
```
/tmp/ipykernel_13824/3984243505.py:53: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=methods, y=accuracies, palette="viridis")
```

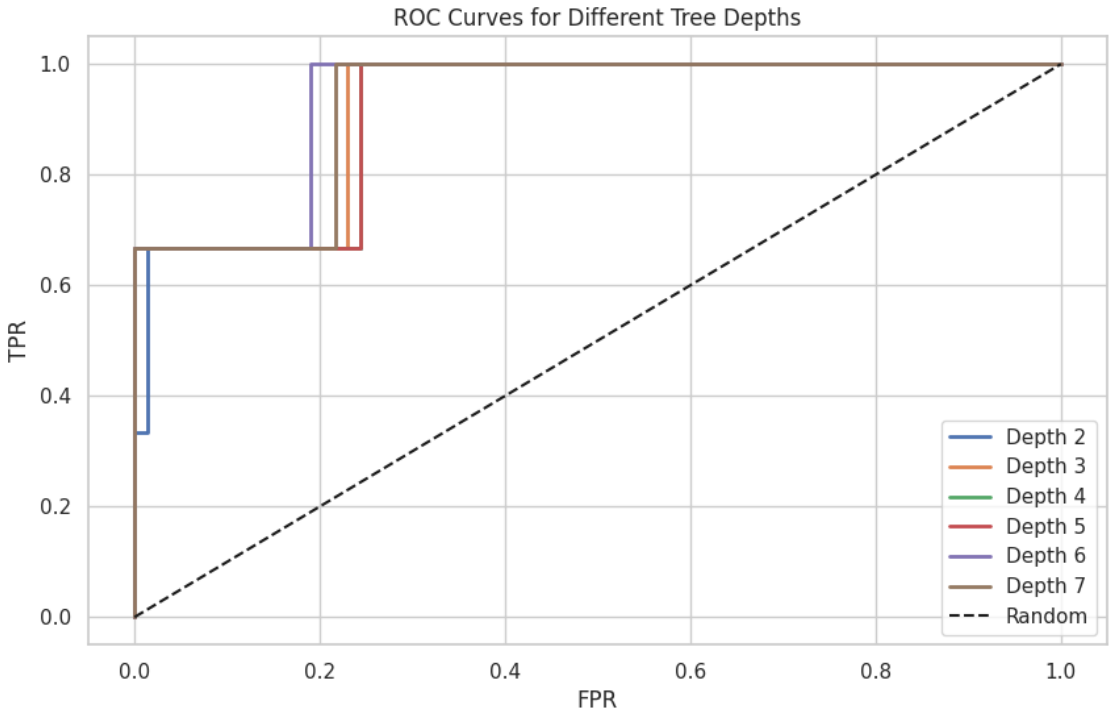


5. Modify the random forest classifier tree depth hyper-parameter for the depth of 2-7 and analyze and comment on the results of the impact of changing the tree depth on the performance (replace the max_leaf_nodes=16 with max_depth = 2 (change from 2-7)).

[15]:
```
depths = list(range(2, 8))
accuracies = []
roc_curves = []

for depth in depths:
```

```
    rf_clf = RandomForestClassifier(n_estimators=500, max_depth=depth,␣
 ↪n_jobs=-1, random_state=42)
    rf_clf.fit(X_train, Y_train)

    rf_y_pred = rf_clf.predict(X_test)
    accuracy = accuracy_score(Y_test, rf_y_pred)
    accuracies.append(accuracy)

    rf_y_prob = rf_clf.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(Y_test, rf_y_prob)
    roc_curves.append((fpr, tpr))

plt.figure(figsize=(10, 6))

# Plot accuracy vs depth
plt.plot(depths, accuracies, marker='o')
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
plt.title('Random Forest Accuracy vs Tree Depth')
plt.grid(True)
plt.show()

# Plot ROC curves
plt.figure(figsize=(10, 6))
for i, depth in enumerate(depths):
    fpr, tpr = roc_curves[i]
    plt.plot(fpr, tpr, linewidth=2, label=f'Depth {depth}')

plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC Curves for Different Tree Depths')
plt.legend()
plt.grid(True)
plt.show()
```


Random Forest Accuracy vs Tree Depth
Accuracy
0.985
0.980
0.975
0.970
0.965
0.960
2
3
4
5
6
7
Max Depth


ROC Curves for Different Tree Depths
TPR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
FPR
Depth 2
Depth 3
Depth 4
Depth 5
Depth 6
Depth 7
Random

[16]:
```
print("\nAccuracy scores for different tree depths:")
for depth, accuracy in zip(depths, accuracies):
    print(f"Max Depth {depth}: {accuracy:.4f}")

optimal_depth = depths[np.argmax(accuracies)]
print(f"\nOptimal tree depth: {optimal_depth} with accuracy: {max(accuracies):.
 ↪4f}")
```

```
Accuracy scores for different tree depths:
Max Depth 2: 0.9610
Max Depth 3: 0.9610
Max Depth 4: 0.9610
Max Depth 5: 0.9610
Max Depth 6: 0.9740
Max Depth 7: 0.9870

Optimal tree depth: 7 with accuracy: 0.9870
```

Random Forest is highly stable, with nearly identical accuracy scores. The dataset is quite small, limiting the overall variability and resulting in step-like ROC curves. The ROC curve is above the random guessing line indicating a good model, however the curve doesn't hug the top left corner meaning there is some uncertainty in the predictions.

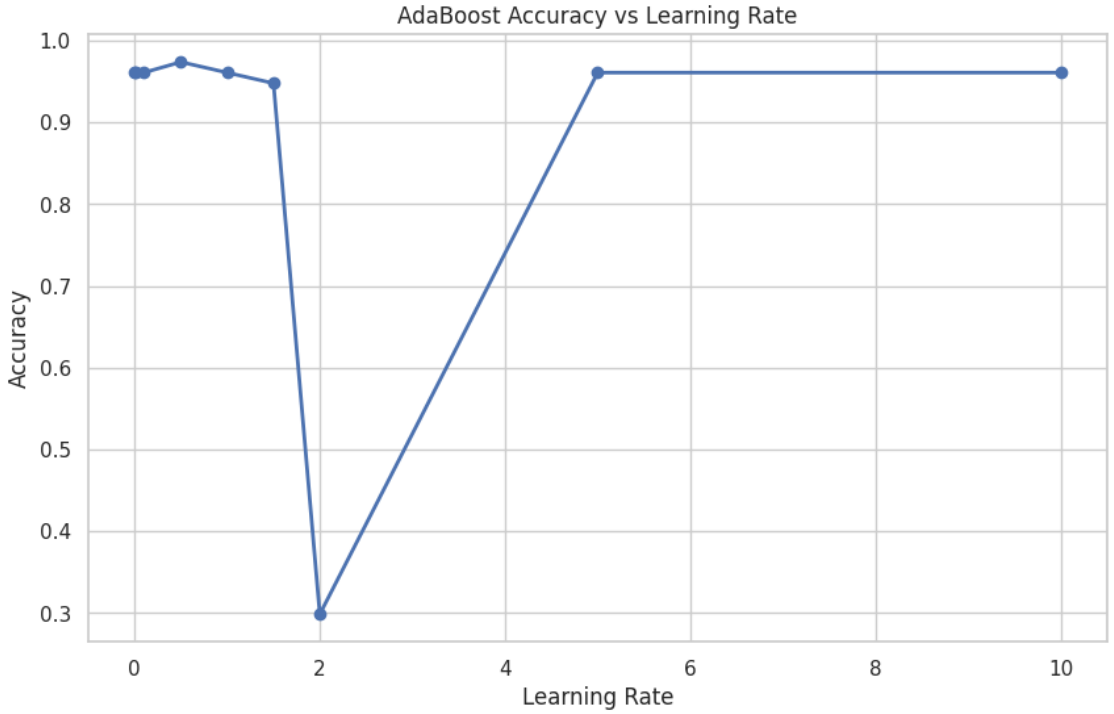
0.1.2 While the most optimal tree depth is 7, a tree depth of 2 results in only 2% less accuracy. Therefore, to combat overfitting and leverage the benefits of an ensemble model, the chosen optimal tree depth is 2.

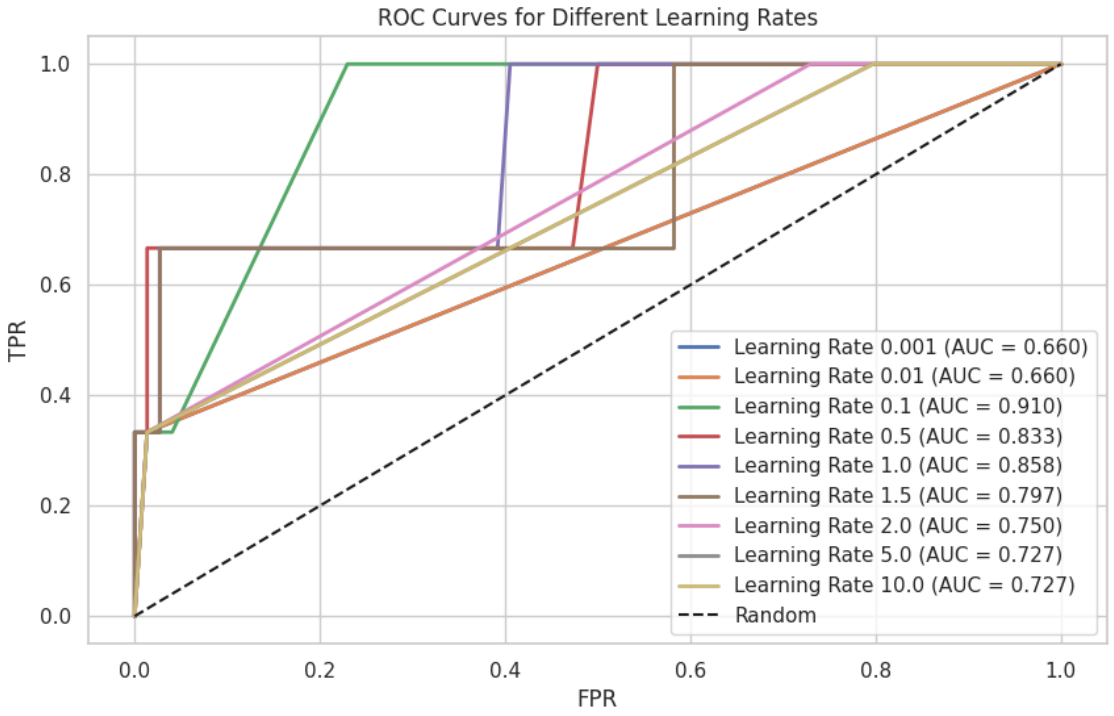
6. For the Adaboost classifier, modify the learning rate to a higher rate and a low rate and analyze and comment on the results (you need to experiment with the learning rate to figure out what range makes sense).

[17]:
```
rates = [0.001, 0.01, 0.1, 0.5, 1.0, 1.5, 2.0, 5.0, 10.0]
accuracies = []
roc_curves = []
auc_scores = []
from sklearn.metrics import roc_auc_score
for rate in rates:
    ab_clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1,␣
 ↪random_state=42), n_estimators=50, learning_rate=rate, random_state=42)
    ab_clf.fit(X_train, Y_train)

    ab_y_pred = ab_clf.predict(X_test)
    accuracy = accuracy_score(Y_test, ab_y_pred)
    accuracies.append(accuracy)

    ab_y_prob = ab_clf.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(Y_test, ab_y_prob)
```

```
    roc_curves.append((fpr, tpr))

    auc = roc_auc_score(Y_test, ab_y_prob)
    auc_scores.append(auc)

plt.figure(figsize=(10, 6))

# Plot accuracy vs depth
plt.figure(figsize=(10, 6))
plt.plot(rates, accuracies, marker='o', linewidth=2)
plt.xlabel('Learning Rate')
plt.ylabel('Accuracy')
plt.title('AdaBoost Accuracy vs Learning Rate')
plt.grid(True)
plt.show()

# Plot ROC curves
plt.figure(figsize=(10, 6))
for i, lr in enumerate(rates):
    fpr, tpr = roc_curves[i]
    plt.plot(fpr, tpr, linewidth=2, label=f'Learning Rate {lr} (AUC =␣
 ↪{auc_scores[i]:.3f})')

plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC Curves for Different Learning Rates')
plt.legend()
plt.grid(True)
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```


AdaBoost Accuracy vs Learning Rate
Accuracy
Learning Rate
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
10


ROC Curves for Different Learning Rates
TPR
FPR
Learning Rate 0.001 (AUC = 0.660)
Learning Rate 0.01 (AUC = 0.660)
Learning Rate 0.1 (AUC = 0.910)
Learning Rate 0.5 (AUC = 0.833)
Learning Rate 1.0 (AUC = 0.858)
Learning Rate 1.5 (AUC = 0.797)
Learning Rate 2.0 (AUC = 0.750)
Learning Rate 5.0 (AUC = 0.727)
Learning Rate 10.0 (AUC = 0.727)
Random
0.0
0.2
0.4
0.6
0.8
1.0

[18]:
```
print("\nAccuracy scores for different learning rates:")
for lr, accuracy in zip(rates, accuracies):
    print(f"Learning Rate {lr:.3f}: {accuracy:.4f}")
```

```
Accuracy scores for different learning rates:
Learning Rate 0.001: 0.9610
Learning Rate 0.010: 0.9610
Learning Rate 0.100: 0.9610
Learning Rate 0.500: 0.9740
Learning Rate 1.000: 0.9610
Learning Rate 1.500: 0.9481
Learning Rate 2.000: 0.2987
Learning Rate 5.000: 0.9610
Learning Rate 10.000: 0.9610
```

AdaBoost shows consistent result for varying learning rates. It strongly indicates that the small dataset contains very clear decision boundaries that are easily captured by the AdaBoost classifier, making the learning rate essentially irrelevant for this particular classification task.

0.1.3 Because of similar accuracies, we evaluate the AUC score to determine the optimal learning rate of 0.1. This achieves the highest accuract and highest AUC score of 0.910

0.1.4 Because most learning rates produce similar accuracies, our dataset may be class imbalanced.

7. Compare the performance of all models (all in steps 2-4) once again this time using cross-validation. Analyze the results and compare with the manual approach (steps 2-4)

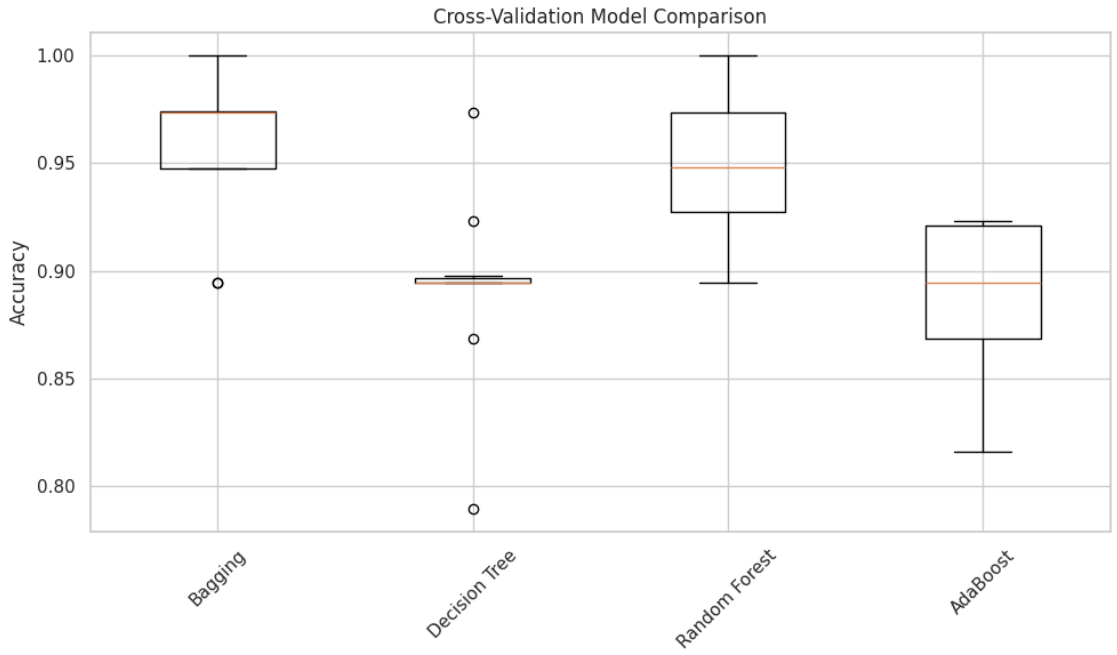
[19]:
```
def evaluate_each_model_in_turn(models, X, Y):
    results = []
    names = []
    scoring = 'accuracy'

    # Perform k-fold validation for each model
    for name, model in models.items():  # Changed from models to models.items()
        kfold = KFold(n_splits=10, random_state=42, shuffle=True)
        cv_results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
        results.append(cv_results)
        names.append(name)
        print(f"{name}: {cv_results.mean():.4f} (±{cv_results.std()*2:.4f})")

    # Create boxplot comparison
    plt.figure(figsize=(10, 6))
    plt.boxplot(results, labels=names)
    plt.title('Cross-Validation Model Comparison')
    plt.ylabel('Accuracy')
    plt.xticks(rotation=45)
```

```
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    return dict(zip(names, [{'mean': r.mean(), 'std': r.std()} for r in␣
 ↪results]))
```

[20]:
```
models = models_to_compare_in_step_7

# Use the function with your existing classifiers
cv_results = evaluate_each_model_in_turn(models, X, Y)

# Compare with previous non-CV results
comparison_df = pd.DataFrame({
    'Non-CV Accuracy': {
        'Decision Tree': dt_accuracy,
        'Bagging': bag_accuracy,
        'Random Forest': rf_accuracy,
        'AdaBoost': ab_accuracy
    },
    'CV Mean Accuracy': {name: results['mean'] for name, results in cv_results.
 ↪items()},
    'CV Std Dev': {name: results['std'] for name, results in cv_results.items()}
})

print("\nComparison with previous non-CV results:")
print(comparison_df.round(4))
```

```
Bagging: 0.9554 (±0.0669)
Decision Tree: 0.8926 (±0.0866)
Random Forest: 0.9476 (±0.0667)
AdaBoost: 0.8899 (±0.0665)
```

```
/tmp/ipykernel_13824/3925844111.py:16: MatplotlibDeprecationWarning: The
'labels' parameter of boxplot() has been renamed 'tick_labels' since Matplotlib
3.9; support for the old name will be dropped in 3.11.
  plt.boxplot(results, labels=names)
```



Comparison with previous non-CV results:

	Non-CV Accuracy	CV Mean Accuracy	CV Std Dev
Decision Tree	0.9091	0.8926	0.0433
Bagging	0.9610	0.9554	0.0335
Random Forest	0.9740	0.9476	0.0333
AdaBoost	0.8961	0.8899	0.0333

1 EDA Overview:

- Non-categorical or numeric data was omitted.
- Log transformation applied to funding amounts to address right-skewed distribution.
- Years of operation dervices from operation years, untransformed (normally distributed).
- Sector was one-hot encoded as an input variable.
- Most features have little to no correlation and the correlation values range from -0.2 to 0.3

2 Output variable selection

- Overhype was chosen due to it's predicability and business relevance.
- Due to the limited count of Overhyped startups, the models were able to accuraty determine pa
- A heatmap is produced to show performance of our models on various output variables, confirmi

3 Model Analysis

- Random Forest performed the best with a 97.4% accuracy, confirmed with the highest CV of 94.8

- Cross-validation confirmed bagging classifiers, decisiontrees, and adaboost performed strongl
- Increasing depth of Random Forest to 7 did not produce significantly greater accuracy
- ROC curve and AUC score were beneficial in determining AdaBoost learning rate as similar accu
- Low standard deviations of CV (3.3-4.3%) indicate stable model performance (low variability)