NUS Fintech Society

Crowdfunding Analysis

Joel, Chong Sun, Jeffrey, Alyssa, Shi Ying

18th May 2025

Introduction

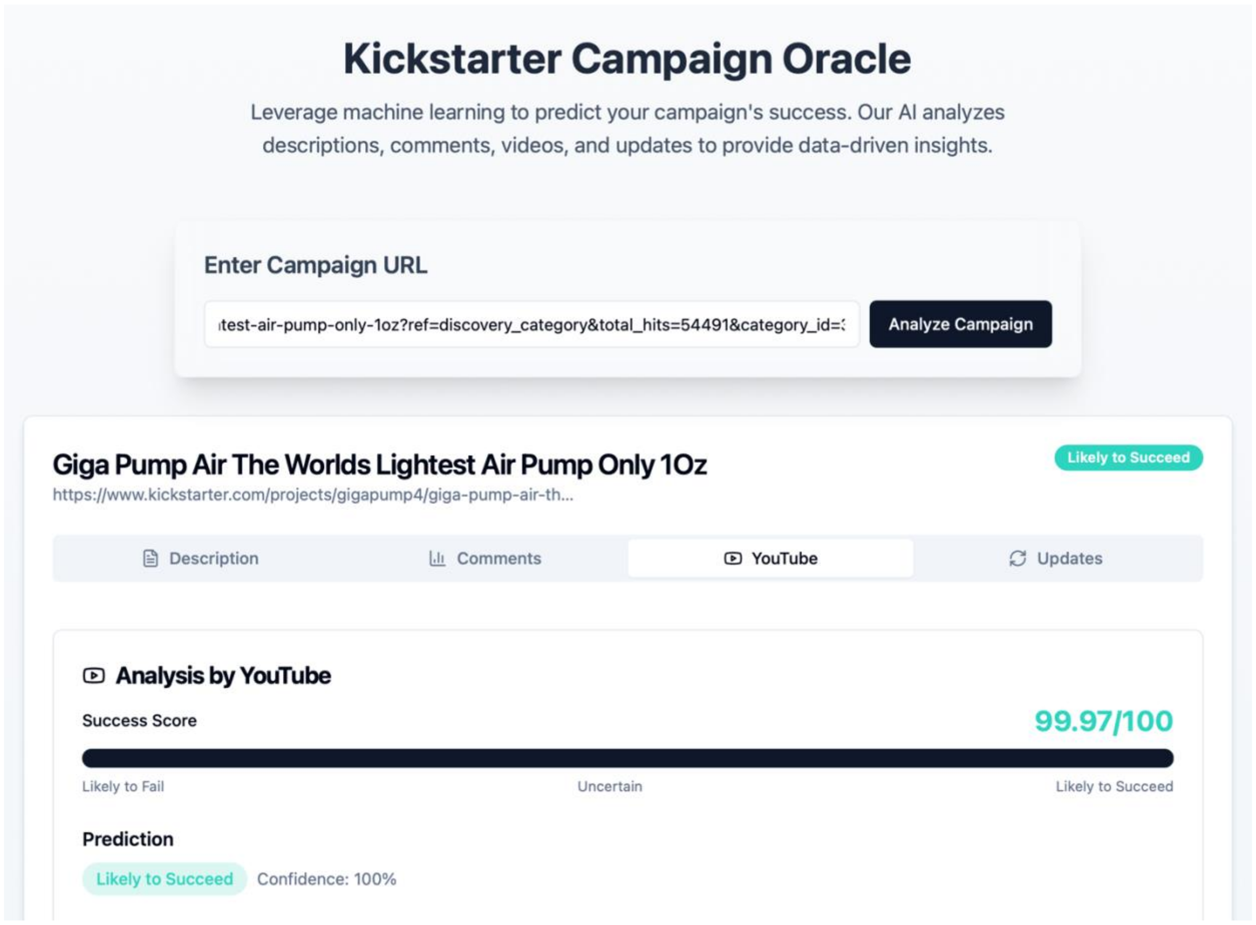
Despite the transformative role of crowdfunding platforms like Kickstarter in launching creative projects, **predicting whether a campaign will succeed remains a significant challenge**. Most people—including creators and backers—struggle to forecast outcomes accurately due to the inherent uncertainty of consumer behavior, limited pre-launch information, and the complex interplay of various campaign factors. This unpredictability often leads to inefficient resource allocation and missed opportunities, emphasizing the need for better predictive tools and strategies.

Existing literature and existing solutions focus on analyzing campaigns via quantitative metrics—number of backers, pledge amounts, funding velocity, etc.—but these methods can’t capture the full story. While early pledge momentum and total backers strongly correlate with final funding, they overlook the rich narrative and community signals embedded in a campaign’s text and updates. For instance, Mollick (2014) shows that “personal networks and underlying project quality” are just as predictive of success as funding‑rate metrics, suggesting that purely numerical models omit key drivers of backer behavior. A semantic text‑analytics study went further, demonstrating that “combining topical features with common numerical features” improved success‑prediction accuracy by over 10% compared to numbers‑only models. And in the most recent work, a multimodal model incorporating text, video metadata, and backer comments achieved state-of-the-art performance—underscoring the added value of qualitative signals alongside traditional KPIs. Yet despite these advances, crowdfunding platforms still surface only basic dashboards of backer counts and pledged sums, leaving creators without the deeper textual insights that could guide more effective storytelling and outreach.

As such our team have decided to create a tool to help people with the analysis of the success probability for Kickstarter Campaigns. In particular, we aim to build the website such that it would scrape live information from various sources and analyse it for them. On top of that, we decided to focus on a mixture of qualitative and quantitative metrics, emphasizing on textual data, rather than solely on numerical metrics. We believe that textual data (if properly trained and cleaned) would be able to provide more information our models can extract which could in turn open up more insights into how well the campaign is doing. Our tool would analyse the following aspects of the campaign:

1. Campaign Description
2. Campaign Comments (within Kickstarter platform)
3. Campaign Updates (within Kickstarter platform)
4. YouTube Comments (of the campaign)

For ease of development, we decided to split up the analysis of the different aspects of a campaign into individual models and datasets which would then be combined with each other in the final website. The following parts of the reports will be structured as a week by week run-down of the work done for each individual campaign aspect:



Kickstarter Campaign Oracle

Leverage machine learning to predict your campaign's success. Our AI analyzes descriptions, comments, videos, and updates to provide data-driven insights.

Enter Campaign URL

ıtest-air-pump-only-1oz?ref=discovery_category&total_hits=54491&category_id=

Giga Pump Air The Worlds Lightest Air Pump Only 1Oz

https://www.kickstarter.com/projects/gigapump4/giga-pump-air-th...

Likely to Succeed

Analysis by YouTube

Success Score 99.97/100

Prediction

Likely to Succeed Confidence: 100%

Website Preview

(to see it in action, go to the [GitHub Repo]() and download the files to run it locally)

Building a Success–Prediction Model Focused on Campaign Updates

When I started, my goal was to harness the “updates” section of Kickstarter projects to predict campaign success using machine learning. What follows is a week-by-week journey—failures, pivots, breakthroughs, and the result.

Week 5:

During Week 5, I focused on learning how to scrape data properly by mastering CSS selectors to target specific elements on web pages. In parallel, I researched sentiment analysis models that could later be used to present campaign insights. However, when attempting to scrape Kickstarter’s technology section, Cloudflare’s anti-bot protection blocked my efforts, which meant I couldn’t progress in that part of the project.

Week 6:

In Week 6, I managed to bypass the initial Cloudflare error, but Kickstarter still prevented me from rendering data while scraping the updates and comments sections from projects. I pivoted by brainstorming alternative features to present if live scraping continued to be unreliable. To maintain progress, I decided to utilise the online (pre-scraped) Kickstarter datasets. I then cleaned one of these datasets to assess which features could be practically implemented. Additionally, I compiled a list of websites that could serve as alternative data sources in case direct scraping remained ineffective.

Recess Week:

Over the recess week, significant breakthroughs occurred. I successfully bypassed Cloudflare by switching to an undetected-chromedriver combined with browser emulation, implementing human-like behaviour such as random delays, mouse movements, and scrolling (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/update-analysis/utils/browser_utils.py). I also incorporated project funding amount tracking to verify accurate sorting of technology projects. With these improvements, I managed to scrape all details from the updates section (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/update-analysis/scrapers/update_scraper.py) using a range of selectors—cumulatively scraping data from 100 projects. Moreover, a link scraper was implemented to harvest project links from the technology section, and I developed robust data cleaning utilities to handle duplicate entries and preserve data quality.

Week 7:

During Week 7, my data collection process was scaled up significantly. I successfully scraped 80 additional projects, which helped refine my scraping techniques and provided more data for feature exploration and modelling.

Week 8:

In Week 8, my efforts ramped up further as I scraped 130 additional projects. I attempted to initiate a Random Forest model but soon encountered issues due to inconsistent metrics—especially around backer numbers and determining whether a project had met its goal. In response, I modified my CSS selectors to accurately capture essential campaign details as well such as backers count, funding period, funding goal, and the actual amount pledged. This refinement allowed me to successfully scrape 170 projects that ended successfully, substantially improving the quality of my dataset for predictive modelling.

Week 9:

During Week 9, I concentrated on projects that had ended but failed. I scraped 170 such projects from an online dataset and implemented a Random Forest model (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/Jeffrey-Branch/analysis/update_analysis.py), achieving a baseline accuracy of 70%. This work provided a reference point for further model improvements in subsequent weeks.

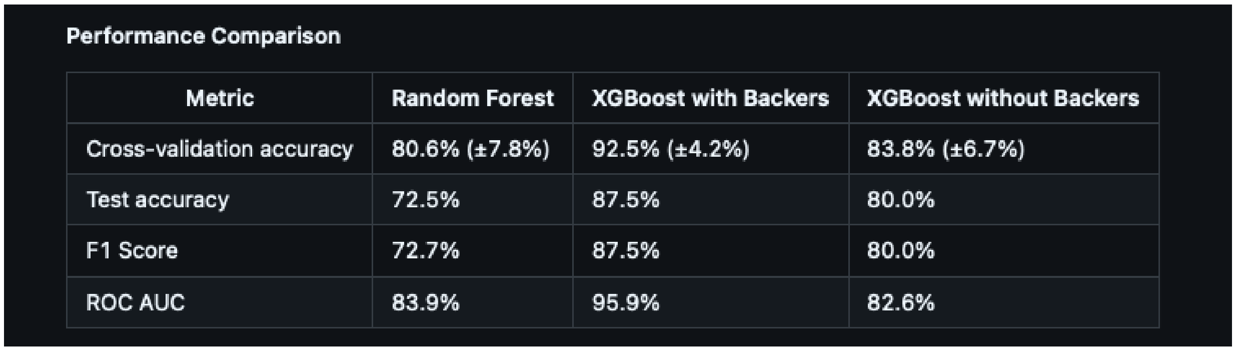
Week 10:

Week 10 was pivotal. I discovered that Kickstarter did not preserve updates for projects that ended in failure which led to a class imbalance in the original dataset. Consequently, I re-scraped live projects solely based on the current funding goal and pledged amount to establish a viable success/fail metric. I rescraped 200 projects (https://github.com/joelleoqiyi/crowdfunding-analysis/tree/Jeffrey-Branch/webscraper/scrapers/scraped_data) and re-implemented my predictive model. Although initial results showed 90% accuracy, overfitting soon became evident. To address this, I scraped an additional 58 live projects from the technology section, retested the model, and removed overly influential backer variables. This led to a more realistic model accuracy of approximately 70%.

Week 11:

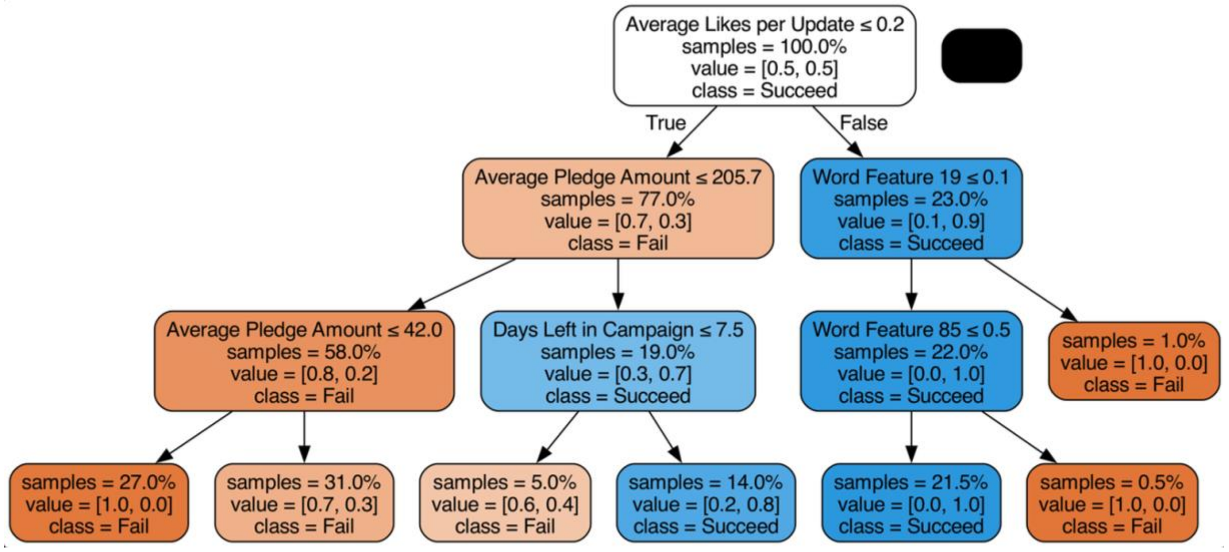
In Week 11, I transitioned to using an XGBoost model as my final predictive model (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/update-analysis/update_analysis_2.py). This model, which achieved an accuracy of 80%, was further fine-tuned and then converted into a PKL file for efficient loading. The fine-tuned XGBoost model was merged into the main branch of the repository, paving the way for smooth integration into my website for real-time predictions.

Results/Findings:



Performance Comparison

Metric	Random Forest	XGBoost with Backers	XGBoost without Backers
Cross-validation accuracy	80.6% (±7.8%)	92.5% (±4.2%)	83.8% (±6.7%)
Test accuracy	72.5%	87.5%	80.0%
F1 Score	72.7%	87.5%	80.0%
ROC AUC	83.9%	95.9%	82.6%



Backend Developments:

On the backend, I helped to implement a Flask API server (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/backend/server.py) featuring prediction endpoints (/predict, /scrape) and integrated the fine-tuned XGBoost model for update analysis. I built out robust model loading and prediction functions with proper error handling with diverse campaign scenarios for testing purposes. Cross-origin resource sharing (CORS) handling was also set up to ensure smooth interactions with the frontend.

Frontend Developments:

On the frontend, an apiClient.ts module (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/frontend/src/utils/apiClient.ts) was developed to facilitate communication between the website and the Flask API. UI components were refined to display prediction results accurately and consistently with the model's actual outputs.

Configuration Improvements:

Configuration-wise, I added necessary dependencies to the requirements.txt file. In addition, minor code corrections were made in multiple files across the project to resolve errors and enhance overall project stability, ensuring a smoother development and deployment process.

Week 12:

During Week 12, my efforts shifted to the frontend. I dedicated time to fixing UI issues and addressing inaccuracies in the displayed prediction information. The integration work was completed so that the model predictions shown on the website accurately match the actual output from the XGBoost model. This full integration ensures that the entire system—from data scraping and preprocessing to prediction and UI display is complete.

Building a Success-Prediction Dashboard Focused on Campaign Description

Our aim was to create an interactive dashboard that allows users to input a rough Kickstarter campaign idea and receive predictive feedback on its likelihood of success. This involved modelling campaign performance based on both textual and numerical features, integrating the results into a usable Streamlit interface, and styling it to fit into our existing website. Below is a summary of our weekly progress:

Week 5:

I began by supporting early data collection efforts, helping to scrape Kickstarter pages 1–50 to aid Alyssa's dataset building as I was individually unable to web-scrape successfully. Simultaneously, I explored potential machine learning directions for the project by reviewing sentiment analysis and time series forecasting models – including BERT, VADER, TextBlob, Prophet, ARIMA, and LSTM. I also studied how ensemble models like Random Forests and Gradient Boosting could simulate A/B testing scenarios using campaign variables such as reward tiers and update frequency.

Week 6 and Recess Week:

During the recess week, I shifted to dataset preparation since Chong Sun and Jeffrey were successful in their web-scraping and I had past Kickstarter datasets that Chong Sun had found to work with. I selected and cleaned key numerical and categorical features, including usd_pledged, goal, fx_rate, staff_pick, category, and currency. I handled missing values, normalized time formats, extracted nested dictionary fields (e.g. campaign category names), and developed scripts to skip over incompatible datasets. At this point, Alyssa and I began considering a pivot from time series forecasting to classification using campaign metadata and text.

Week 8:

I incorporated Alyssa's OpenAI-powered description generator into my dataset pipeline, generating more robust campaign descriptions from blurbs. I also attempted to scrape additional projects using Jeffrey's randomized mouse-clicker method but was ultimately unsuccessful with the scraping. However, with at least the preliminary data still in place, I began experimenting with model training, preparing for a working version.

Week 9:

I attempted to implement LightGBM and XGBoost models but ran into roadblocks related to sentiment preprocessing and environment setup (e.g. missing OpenMP on macOS). I explored sentiment analysis on campaign blurbs using VADER and began matching project names to Chong Sun's scraped comment dataset. I also learnt an interesting point raised during the meeting about the need for the sentiment analysis team to clean out manager-authored comments, which biased sentiment positively. The week concluded with a decision to merge multiple sentiment sources for training a unified model.

Week 10:

This week marked a pivot towards explainability. I conducted exploratory data analysis using decision tree classifiers and logistic regression (models that have higher explainability) and produced a final pipeline detailing how I went about deciding feature importance, feature selection and finally training a decision model to determine success thresholds for the most important textual and numerical features (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/Shi-Ying-Branch/eda_decisiontree.ipynb). These models helped me identify the most impactful text keywords and numerical thresholds (e.g. funding goal, backer count, FX rate) for classifying campaign success or failure. I also studied how decision trees work, particularly how Gini Impurity governs feature splits, and used these insights to interpret model outputs and guide feature selection.

Week 11 - 12:

I finalized a functional prototype of the Streamlit dashboard (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/Shi-Ying-Branch/kickstarter_tree_dashboard.py). Users can now enter a rough campaign idea (via campaign name, category, and summary) and receive a prediction of campaign success based on model-generated insights. Keyword suggestions are drawn from a TF-IDF + logistic regression model, and numeric guidance (e.g. recommended funding goal or backer count) is based on thresholds extracted from the decision tree. I fine-tuned the model by removing overly deterministic features like percent funded and achieved strong performance across various test scenarios.

Results/Findings:

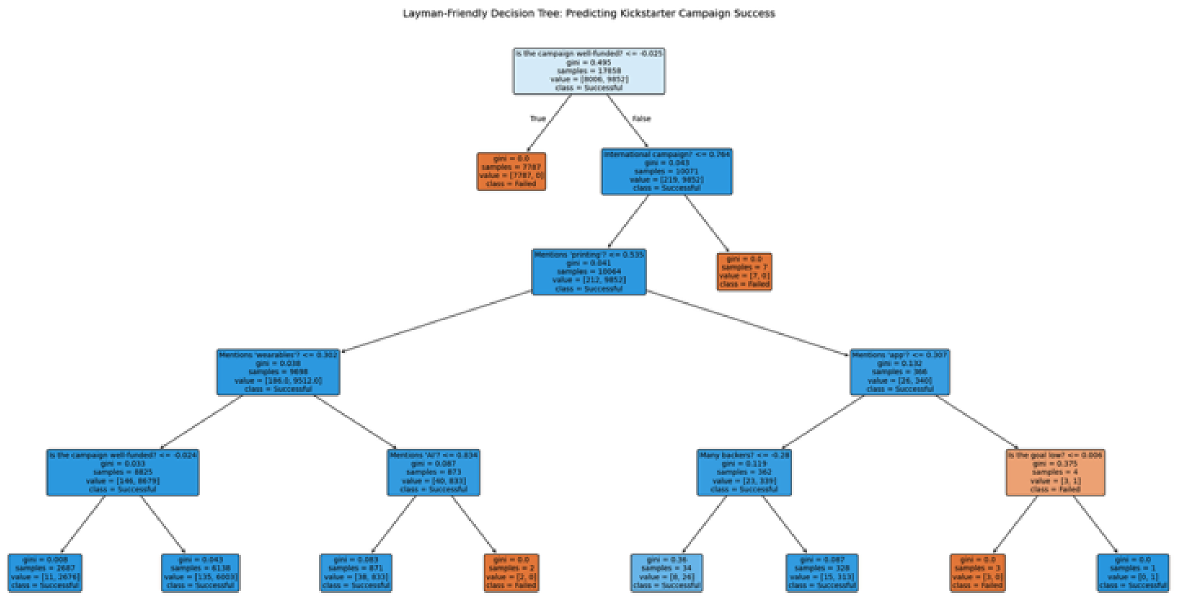
Model	**Accuracy**
Simple Decision Tree	0.59
Logistic Regression (with improvements, only textual features)	0.73
Logistic Regression (with both textual and numerical features)	0.84

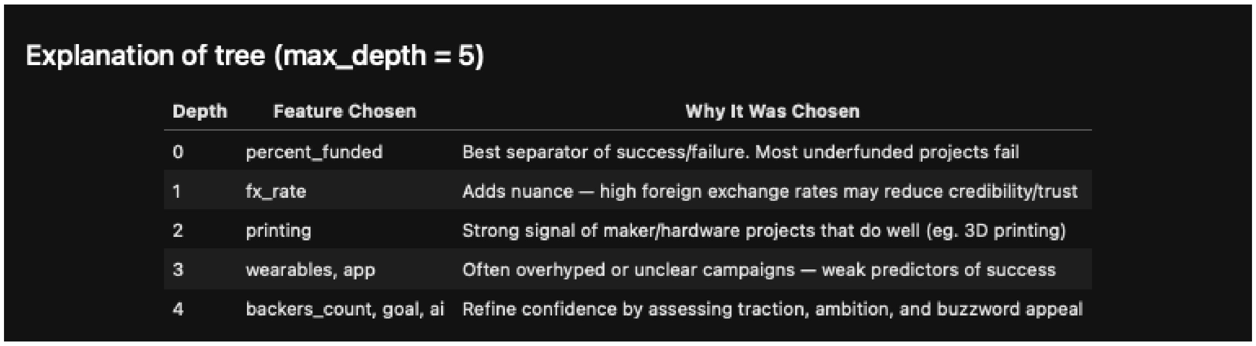
SUCCESS-PREDICTING KEYWORDS:
Words like "hardware", "gadgets", "pi", "usb", "arduino", and "learn" show strong positive log-odds, meaning their presence significantly increases the likelihood of campaign success. These words are often associated with:

- Tangible, buildable products
- DIY/maker culture (eg. Arduino, Raspberry Pi)
- Educational or functional use cases
- Clear utility and real-world application

FAILURE-PREDICTING KEYWORDS:
Words like "software", "apps", "electronics", and "wearables" have negative coefficients, indicating they are more commonly found in unsuccessful campaigns. These may suggest:

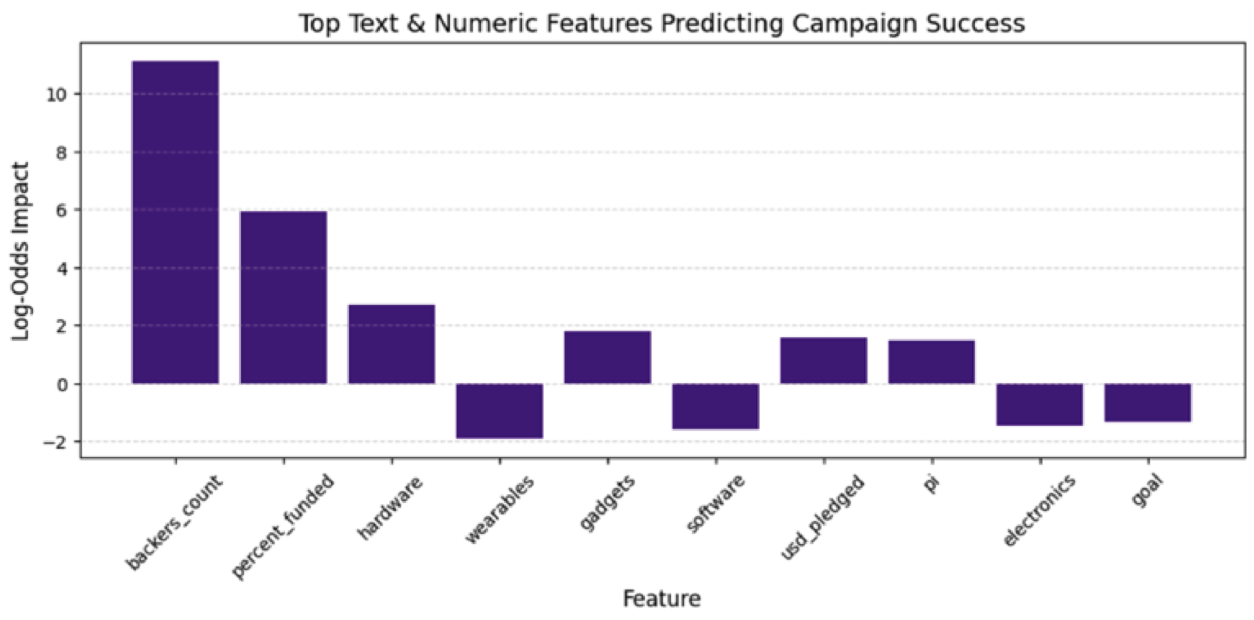
- Overcrowded or saturated markets (apps/software)
- Vague or less tangible ideas
- Concepts that are harder to convey value clearly (eg. abstract software or generalized "electronics")





Explanation of tree (max_depth = 5)

Depth	Feature Chosen	Why It Was Chosen
0	percent_funded	Best separator of success/failure. Most underfunded projects fail
1	fx_rate	Adds nuance — high foreign exchange rates may reduce credibility/trust
2	printing	Strong signal of maker/hardware projects that do well (eg. 3D printing)
3	wearables, app	Often overhyped or unclear campaigns — weak predictors of success
4	backers_count, goal, ai	Refine confidence by assessing traction, ambition, and buzzword appeal



Positive = success predictors | Negative = failure predictors (based on log-odds impact).

Week 13:

This week focused on frontend integration. As I used Streamlit (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/description-analysis.py) instead of the React + NodeJS architecture that the rest of the team is using, we decided to combine my work with the rest of the team by linking the 2 websites with a button. As such, I customized the dashboard's UI to align as much as possible with the main website by overriding Streamlit's default styling using Tailwind-inspired CSS. However, styling conflicts with Streamlit's Emotion-based CSS system presented challenges. To integrate with the main website, I had to explore a backend solution that launches both the Flask API and Streamlit dashboard in parallel using Python's subprocess and threading libraries. This would ideally allow users to access the dashboard via a button on the main page. I also learnt about conditional rendering in React that Joel shared during the meeting which would coordinate frontend navigation.

Backend Developments:

On the backend, I ensured the decision tree and logistic regression models were exported as .pkl files and loaded into a Flask-compatible API environment. Thresholds for numerical dials and feature weights for keyword suggestions were handled in a modular fashion to support efficient inference during real-time user input.

Frontend Developments:

I contributed a call-to-action section with a "Text your own campaign" button under the campaign input form. This button is supposed to allow users to test a hypothetical idea without a real Kickstarter URL by opening the Streamlit app in a new tab. I styled the button and message to maintain a cohesive visual experience with the main website's Tailwind CSS design, engaging Alyssa's help to integrate the button on the frontend as I was facing some troubles with it.

Building a Success-Prediction Model Focused on Campaign Comments

Week 5:

The first part of the task was to be able to scrape a lot of data from Kickstarter to train the different models. For me, since I was supposed to predict the amount funded based on the sentiments, this meant that I had to scrape the comment text for each campaign. The idea was:

1. Iterate through the Kickstarter pagination pages, incrementing the page query param in the URL and scraping all the campaign links for each page.
2. Aggregate all the URLs into a single file.
3. Run the comment scraper for each campaign link in the file. The file can be shared across projects to ensure we are using the same data.

I had no prior experience scraping dynamic websites with Selenium and was only familiar with scraping static websites with Beautiful Soup. While Selenium allows one to scrape dynamic websites, to understand its full set of capabilities well, I had to learn about CSS Selectors and X-Paths.

This week was focused on building the link scraper first. However, the first challenge I faced was that Selenium is blocked by Cloudflare, regardless of whether headless or my own chrome profile was used.

Week 6 and Recess Week:

While the code for scraping worked on singular discovery and campaign pages, it did not work when we had to navigate across pages. Thus, these two weeks was spent trying to get the scrapers to work. I also tried to employ Docker containers to get around the Kickstarter's scrapping detection measures and undetectable chromedriver, which is essentially a Selenium wrapper that helps avoid detection. Both methods tried initially failed to get past the detection.

Eventually, with Jeffrey's code, we managed to overcome this issue. While his code did include some features such as randomly moving a mouse, I managed to get the scraper to work without using the random mouse movements, and just the correct undetectable chromedriver settings.

Week 7:

Now that we can scrape the comments, I managed to run the scrapers this week to:

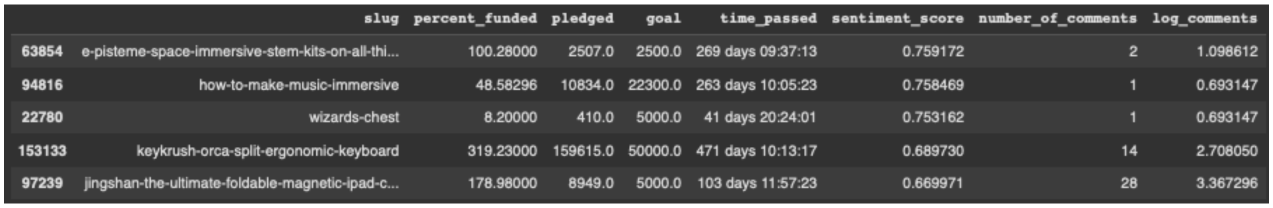
1. Scrape over 200 links from the technology discovery section.
2. Navigate to these links
3. Run the comment scraper on these links and scrape the comments.

The comments were aggregated into a single file for easy import into Google Colab.

Week 8:

For our sentiment analysis model, we decided to use DistilBERT as the model as most comments are relatively short. Simple models may not be able to capture the sentiment as well, and therefore a pre trained large language model was chosen.

The results of the sentiments were averaged for each campaign and this statistic was then passed into simpler machine learning models. However, the sentiments barely showed any correlation to the percentage funded and the success of the campaign. What did show a significant correlation was the number of comments. This made sense from a business perspective, as regardless of whether the comments were good or bad, in general, if people commented it reflects interest in the product which was better than no interest at all.



	slug	percent_funded	pledged	goal	time_passed	sentiment_score	number_of_comments	log_comments
63854	e-pisteme-space-immersive-stem-kits-on-all-thi...	100.28000	2507.0	2500.0	269 days 09:37:13	0.759172	2	1.098612
94816	how-to-make-music-immersive	48.58296	10834.0	22300.0	263 days 10:05:23	0.758469	1	0.693147
22780	wizards-chest	8.20000	410.0	5000.0	41 days 20:24:01	0.753162	1	0.693147
153133	keykrush-orca-split-ergonomic-keyboard	319.23000	159615.0	50000.0	471 days 10:13:17	0.689730	14	2.708050
97239	jingshan-the-ultimate-foldable-magnetic-ipad-c...	178.98000	8949.0	5000.0	103 days 11:57:23	0.669971	28	3.367296

Week 9:

This week, I attempted finetuning the sentiment analysis model. I suspected that perhaps the scores of the model were not accurate because of the lack of fine-tuning on a comment dataset. Hence, I wrote queries and used OpenAI's API to generate comment sentiment pairs to finetune the model. However, the finetuned model did not show any significance performance improvement, and as such we just went ahead with the pretrained model. I also managed to scrape more details about the comments, such as whether the poster of the comment was the creator of the campaign, as sentiments by the creator of the campaign might be biased and not accurately reflect the success of the campaign.

Week 10:

During this week, I discovered that I had accidentally mixed some training and validation examples, which had artificially inflated our R^2 from 0.6 down to a more realistic 0.19 once corrected. By revisiting our feature set—engineering new variables and removing leakage—we were able to raise R^2 back up to 0.48. While this performance is still only moderate, it highlights areas for future work: ensuring airtight data splits, exploring richer features, and perhaps combining comments with other data sources to improve predictive power. It also suggests that Kickstarter comments alone may be too noisy or biased to serve as the sole signal of campaign success.

Week 11:

For the last week, I integrated my scraper and predictor modules into Jeffrey's backend: migrating all relevant files, configuring the dependencies, and wiring the scraping and prediction functions into the application's API layer. Although the current accuracy remains below our target, this integration lays the groundwork for continuous improvement. It further underscores that relying solely on in-platform comments may not capture all the drivers of campaign outcomes—something we can address in future iterations by adding alternative data streams (e.g., updates, external social media) and experimenting with ensemble models.

Building a Success-Prediction Model Focused on YouTube Comments

Week 5 - 6:

Since we are focusing on Tech Campaigns which often collaborate with Youtubers to promote their products in the videos. We identified that a possible avenue for expanding the current literature would be to analyse comments on YouTube for insights regarding the likelihood of a campaign's success, since those comments would likely be the most "truthful" and unfiltered leading to potentially greater accuracy at prediction.

We also explored other avenues like Reddit or X (i.e. Twitter) for data sources, but those APIs are harder to access (or cost money) and do not provide information into past campaigns (eg: X's API only allows query for the past 7 days).

We used Google YouTube API to extract comments information from the relevant YouTube Videos related to the campaign (via the blurb/campaign name). (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/webscraper/youtube_api_scraper_data/data_scraper.ipynb) This process took a little longer than expected due to various reasons such as YouTube API has a quota per day of 10,000 units per day and hence, we scraped 150 campaigns for a start.

Recess Week:

We analysed the comments and realised that each campaign contains a lot of comments (thousands of comments; with some campaigns going up to 80,000 comments each) and running it through our model would take too much computational power which we might not have, as such we need to narrow down the comments by a huge factor, to balance accuracy and inference duration.

We decided to run the comments through a zero-shot BERT classifier for the labels of ["about campaign", "not about campaign"] between the comment and the campaign blurb (gotten from the online dataset of past Kickstarter Campaign) to get a "relevance score" for each comment which would help us to narrow down our comments to those that are the most relevant to the campaign. (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/youtube-training/Kickstart_Campaign_Analysis_Youtube_Comments.ipynb)

We set a tentative threshold for "relevant" comments at 80%.

Week 7:

After narrowing down the relevant comments, we proceeded to finetune a BERT classifier pipeline (“bert-base-uncased”) to classify the comments into successful campaign or failed campaigns. (https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/youtube-training/Kickstart_Campaign_Analysis_Youtube_Comments.ipynb) Using the following settings:

- per_device_train_batch_size=8
- learning_rate=2e-5
- num_train_epochs=3
- weight_decay=0.01

However, the initial results showed an issue where all the predictions would be labelled as “successful” which meant that the model was essentially taking a shortcut and labelling everything the same class i.e. model collapse.

Week 8:

To resolve this, we decided to do the following:

- Unfreeze some layers (initially just the classifier layer) to see if the model can be finetuned better to avoid model collapse issue.
- Unfreeze even more layers (the last 3 layers + classifier layer) to see if can fix the output collapse issue.
- Tried DistilBERT instead since the full BERT might be too complicated for the number of datapoints we have, to see if can fix the overfitting issue.

(https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/youtube-training/Kickstart_Campaign_Analysis_Youtube_Comments.ipynb)

We also started to scrape more datapoints (i.e. for more campaigns) since our dataset might be too small. We ran the datapoints through the pre-processing pipeline of getting their relevance scores. We doubled our dataset to over 300 datapoints.

However, the outcome is still the same where the model collapse to only one label.

Week 9:

Upon further investigation, we realised that our training might have been suffering from class imbalance issues, which we proceeded to fix by augmenting the default unweighted cross-entropy, by passing in class weights so that errors on the minority class carry more penalty.

Furthermore, we suspect that using DistilBERT might still be too complicated for our model to effectively learn features, as such we trained simpler models such as Logistical Regression and Random Forest Classifier on features such as number of comments, mean/median/std of relevance scores, percentage of comments with high relevance, etc.

(https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/youtube-training/Kickstart_Campaign_Analysis_Youtube_Comments.ipynb)

However, the outcome is still the same where the model collapse to only one label.

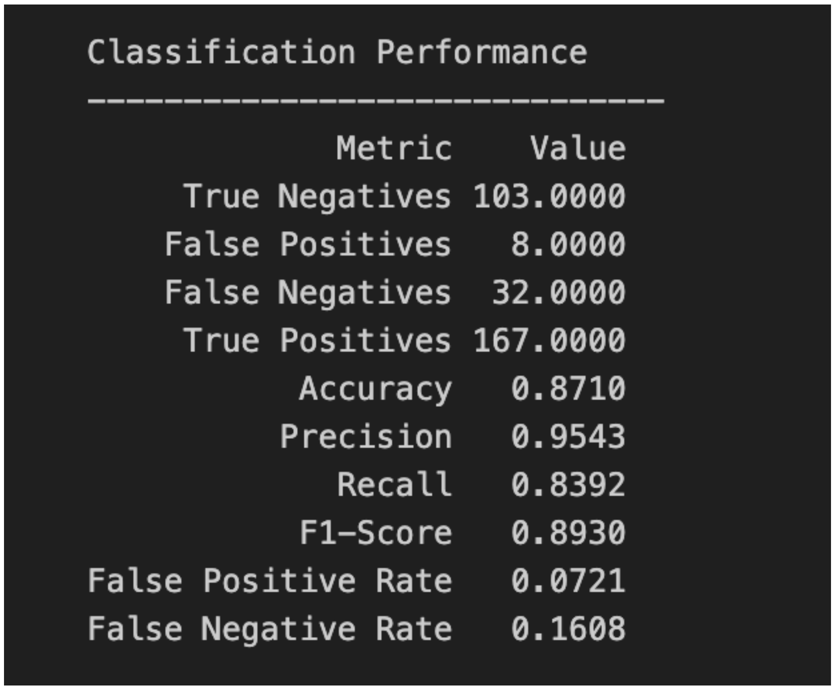
Week 10:

We decided to return to using BERT models since we wanted the nuances of analysing the content of the comments rather than just numerical features of the comments. We decided to narrow down the finetuning to just the top 50 most relevant comments (by score) instead of the original method which uses the first 50-100 relevant comments (that crosses the 80% relevance threshold). We found that this method surprisingly solves the model collapse issue.

Afterwards, we further tweaked the hyperparameters of the training by adjusting the learning rates, number of epochs, weight decay rate, etc to further finetune and improve the model's accuracy. We kept in mind the class imbalance issues faced in the past and continued the use of the Weighted Cross Entropy trainer.

(https://github.com/joelleoqiyi/crowdfunding-analysis/blob/main/youtube-training/Kickstart_Campaign_Analysis_Youtube_Comments.ipynb)

In the end, we managed to achieve an accuracy score of over 87%.



```
Classification Performance
------------------------------
              Metric     Value
      True Negatives  103.0000
     False Positives    8.0000
     False Negatives   32.0000
      True Positives  167.0000
            Accuracy    0.8710
           Precision    0.9543
              Recall    0.8392
            F1-Score    0.8930
 False Positive Rate    0.0721
 False Negative Rate    0.1608
```

Week 11 - 13:

This few weeks was focused on developing the website (https://github.com/joelleoqiyi/crowdfunding-analysis/tree/main/youtube-analysis).

Backend Developments:

We had issues with integrating the model into the backend due to the size of the model and the amount of computational power it takes. We found out that by loading the model onto MPS (Apple's version of GPU), and adjusting the batch sizes in which the various models loaded and ran at, we could reach a stage where the model takes an acceptable amount of inference time while keeping the accuracy.

Frontend Developments:

We built the front end to ensure a sleek user interface which allows the users to quickly and efficiently get the information they require about the success probability of a particular campaign.

Conclusion

Despite the challenges inherent in scraping dynamic sites and juggling multiple data streams, this project demonstrates that combining qualitative text signals with conventional numerical metrics can meaningfully improve campaign-success predictions. Across our four pipelines:

- Campaign Description yielded up to 84 % accuracy when blending TF – IDF features with a logistic regression model.
- Campaign Updates achieved 80 % accuracy with XGBoost after careful handling of class imbalance and feature engineering.
- Kickstarter Comments proved less reliable on their own with its R^2 stalled below 0.5 and model collapse issues persisted, suggesting that in-platform comments may be too sparse or biased for standalone prediction.
- YouTube Comments reached 87 % accuracy once we filtered for relevance and fine-tuned a BERT-based classifier, underscoring the value of off-platform, unfiltered community feedback.

By structuring each aspect into its own model and then integrating them via a unified Flask backend and React/Streamlit frontend, we've built a modular, extensible tool that surfaces deeper insights than pledge-count dashboards alone.

Next Steps/Future Work

Below are some possible extensions of the project for future work:

- Ensemble & Multimodal Fusion
  - Combine the four separate pipelines into a single ensemble model (e.g. stacking or voting) to capture complementary strengths and further boost overall accuracy.
- Time-Series & Engagement Dynamics
  - Incorporate update cadence, comment volume trajectories, and pledge-velocity curves as time-series features. This could reveal early momentum signals that static snapshots miss.
- Broader Data Sources
  - Explore additional external signals-Twitter/X threads, Reddit discussions, Facebook groups, or domain-specific forums-to enrich context beyond YouTube comments and platform updates.
- Robust Scraping & API Use
  - Reduce reliance on brittle web-scraping by leveraging official APIs (where available) or maintaining periodic offline caches. This will stabilize data pipelines and minimize interruptions.
- Cross-Category Generalization
  - Expand beyond the Technology category to validate whether the same text-driven features hold predictive power in art, design, games, or social-impact campaigns.

By pursuing these extensions, we can evolve our prototype into a production-ready forecasting service-one that not only quantifies "chance of success" but also equips creators with actionable, data-backed storytelling guidance.