Final Report
Team A. C. G

Introduction

Background:

We had a rough time at first finding a good project. There were many options out there that could have been considered. Ultimately we decided on Kickstarter as our field of topic due to its social importance and effect on the population.

Motivation:

It is important to study what makes a Kickstarter successful. Kickstarters help everyday people achieve real-life goals that they would not be able to obtain normally without any financial backing.

Some things to consider when making a Kickstarter is when to launch and end it, how much capital to ask for, what the title should be, and what the main and sub-category the Kickstarter is in.

Challenges:

One of the challenges we encountered immediately was the coding. On our first implementation, we created a KNN that ended up getting an accuracy of 99%. We ended up learning later that this was incorrect and determined that some of the columns of data had much higher influence compared to others, which allowed it to get high accuracy.

Another Challenge we had was that the data had a bias as to which Kickstarter failed and which succeeded, 60% of the Kickstarter failed leaving 40% as a success. So we decided to even the data before anything, so it is equally split 50% success and failure.

Summarization of Contributions

Collin kept our group organized and made a plan of action for everyone to do. He also made the initial notebook which we later refactored into a different notebook. He also organized the Report. Greg added comments to the code for clarity. He also transformed the data and worked on formal definitions and equations for the details of the proposed methods. Alec helped with the data processing of our input, such as normalizing the distribution and values, one hot encoding, and sentiment analysis on the titles**.** We each helped implement the many different models inside the notebooks as well as worked on this report. As well as participated in group meetings and discussions for topic choosing. Lastly, we worked on the presentation report, video, and slides.

Method

Formal definition of targeted problems

The problem we were trying to solve was to predict the success or failure of a Kickstarter given features and samples. This can be formally described as a binary classification problem. That would be two primary classes: success and failure. It can be expected that the probability of selecting a class would be 50%, given that there are only two. To determine if our methods are a success we will be using the accuracy and macro F1-score as metrics to compare against our baseline.

Precision = TruePositive / (TruePositive + FalsePositive)
Recall = TruePositive / (TruePositive + FalseNegative)
$F_1 = (2 * P * R)/(P + R)$
Accuracy = (TruePositive + TrueNegative) / ALL

Details about Proposed Methods

Uniform Random Model:

This is a very simple method where the success or failure is determined solely at random. It uses a uniform distribution to select either success or failure. The probability is the same for each sample. This approach requires no training data and serves as our baseline.

KNN Model:

The next model we went to use is a KNN. We thought that this would have been a good choice given that it may put similar Kickstarters closer together in distance, so it could predict success or failure.

KNN Formal Definition:

Test point: x
Define the set of the k nearest neighbors of x as Sx. formally Sx is defined Sx ⊆ D s.t. |Sx| = k and ∀(x',y') ∊ D\Sx',

dist(x,x') ≥ max(x'',y'') ∊ Sx dist(x,x''),

We can then define the classifier h() as a function returning the most common label in Sx:

h(x) = mode({y'' : (x'',y'') ∈ Sx}),

Where mode(.) means to select the label of the highest occurrence.

Minimum distance sum((validation - training)^2)
X = {C,MC,G,D,SM,EM,SD,ED}

$$arg \min_{n=1,\ldots,N} \sum_{d=1}^{D} (X_d - X_{nd})^2$$

Where n is the number of training samples and d is the number of testing samples.

And Knn(X) = {nn1(X),...,nnK(X)} where nn1 < (nn2,...,nnK), nn2 < (nn3,...,nnK),..., nnK-1 < nnK. And nn means “Nearest Neighbor”

Neural Networks:

The last type of model we used is a neural network. We thought that it may be good, as it could potentially “learn” patterns in our data, that could help it predict the results.

We went to creating two models:
Simple Neural Network:
Inputs: 242
Output: 1
Layers:
242 -> 242
LeakyRelu
242 -> 484
LeakyRelu
484 -> 484
LeakyRelu
484 -> 484
LeakyRelu
484 -> 1
Sigmoid

Complex Neural Network

It is not really that far different, it is just complex as a way to identify the two types of neural networks we are testing. This one just has wider layers and more layers.

Inputs: 242
Output: 1
Layers:
242 -> 726
LeakyRelu
726 -> 726
LeakyRelu
726 -> 726
LeakyRelu
726 -> 726
LeakyRelu
726 -> 726
LeakyRelu
726 -> 726
LeakyRelu
726 -> 1
Sigmoid

Neural Networks Formal Definition:

A k-layer neural network is a mathematical function f, which is a composition of multivariate functions: f1, f2, …, fk, and g, defined as:
f: Rn->Rp
f=g o fk o…o f2 o f1

Where n is the dimension of the input x, p is the dimension of the output y, g is the output function, each function f_i is itself a composed multivariate function.

f_i(x) = a(w_i*x+b_i)
f: $R^n -> \ R^p$
f(x) = g o fk o…f2 o f1(x)
fi(x) = a(wi * x + bi)

Each function f_i is a composed function with the functions ‘wx+b’: a linear combination of the input x with its coefficients w, plus a bias b, and ‘a’: that is called activation function.

The activation function can take various forms such as sigmoid, tanh, ReLU, etc.

Sigmoid: $\boldsymbol{\sigma}(x) = 1/(1+e^{-x})$
Leaky ReLU: $f(x) = max(0.01 * x, x)$

Experiment Details

Data:

Our Data consisted of all Kickstarter between 2009 and 2018. As for its columns, it has an ID, name, subcategory, main category, currency, start, end, goal, pledged, state, number of backers, country, USD pledged, USD pledged real, USD goal amount, USD goal amount real. There were about 325k Kickstarters in total. Since we are doing classification our most important column is state. It shows the current state of the Kickstarter and labels it as successful, failed, ongoing, suspended, and canceled. We will be focusing solely on success or failure.

Metrics:

Below is a subset of metrics of our data. This is done per category and shows some of the important details. For example: the number of successes, failures, and the ratio, the average duration in days, and average goal in USD.

Category	% of data	Success vs Failure	Average Duration (days)	Average Goal (USD)
Publishing	11%	12300 vs 23113 Success rate: 35%	33.7	$17,555
Film & Video	175	23612 vs 32891 Success rate: 42%	35.0	$76,122
Music	14%	24105 vs 21696 Success rate:	35.3	$14,449

		53%		
Food	7%	6085 vs 15969 Success rate: 28%	33.8	$48,167
Crafts	2%	2115 vs 5703 Success rate: 27%	31.4	$9,954
Games	9%	12518 vs 16002 Success rate: 44%	32.2	$38,947
Design	8%	10549 vs 14814 Success rate: 42%	34.1	$38,238
Comics	3%	5842 vs 4036 Success rate: 59%	33.8	$20,192
Fashion	6%	5593 vs 14181 ratio Success rate: 28%	32.6	$21,885
Theater	3%	6534 vs 3708 Success rate: 64%	33.3	$23,133
Art	8%	11510 vs 14130 Success rate: 45%	32.3	$39,432
Photography	3%	3305 vs 6384 Success rate: 34%	33.5	$11,098

Technology	8%	6433 vs 20613 Success rate: 24%	35.2	$97,964
Dance	1%	2338 vs 1235 Success rate: 65%	32.8	$8,332
Journalism	1%	1012 vs 3136 Success rate: 24%	34.2	$70,111

Data Preprocessing:

We had to do a lot of preprocessing on this data before using it. The first thing we did was remove some of the unnecessary columns. We got rid of id, pledged, USD pledged, USD pledged real, goal, and number of backers. We got rid of these columns, primarily because they include data that would determine the success or failure. We chose to only use data that would be available as if the Kickstarter had just launched when making our prediction. The final columns we ended up using, before applying other further transformations, were: name, category, main category, currency, country, state, USD goal amount, and the timestamp of launch and deadline.

After removing the unnecessary columns, further transformations were made. The next thing we did was use the launch and deadline timestamp to generate new columns. These were: duration (in days), start month (name), end month (name), start year, and end year. Now the initial timestamp columns are no longer needed.

The next thing we did was run a sentiment analysis on the data. Initially, we decided not to include the title column for a Kickstarter, however, we felt like it should be included in the features in some way. After running the sentiment analysis it added four columns: how negative, how positive, how neutral, and compound of all three values. After this, there was no need for the original title. The next thing we did was one hot encoded some of the categorical columns, which were columns: category, main category, currency, country, start month, and end month. This was done to not imply a relationship between two, where if one was value 1 and another was value 2, category 2 was not higher than category 1.

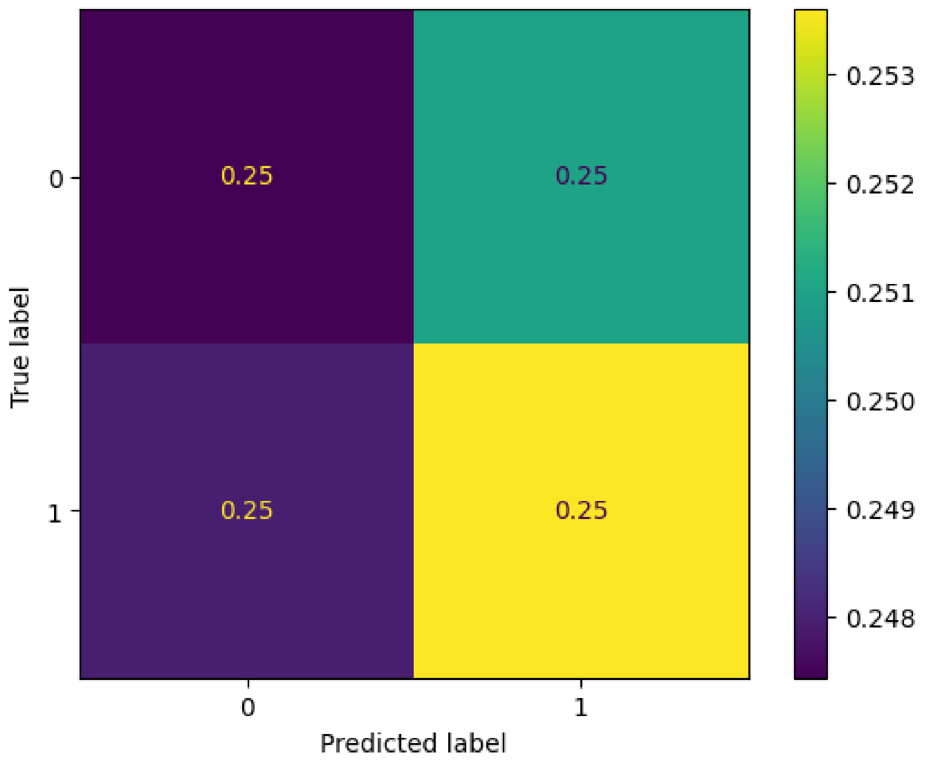
The last thing we did was normalize some of the columns. We noticed that normalizing the categories resulted in an improvement in the KNN. For the duration and goal, we used z-score normalization. We also used min-max normalization for the start and end year.

After all of this, we split the data into three categories: training (80%), testing (10%), and verification (10%). The training data is used for training. The testing data is used primarily by the neural network to monitor the progress of its training. Lastly, the verification data is shared among all models as a way to compare them against each other.

Baseline:

For our experiment, we will be using the random model as our baseline. It represents a scenario in which each Kickstarter is independent of each other. The results are:

Class	Precision	Recall	F1-Score	Support
0 (Failure)	0.50	0.50	0.50	13344
1 (Success)	0.50	0.51	0.50	13427
Accuracy			0.50	26771
Macro Average	0.50	0.50	0.50	26771
Weighted Average	0.50	0.50	0.50	26771



The results were generated using the validation set, which will be the same one for all other models, as a way to compare each of them with the same unseen data. As expected the accuracy is 50% and the macro F1-Score is also 50%. The chart shows the prediction versus truth distribution, as seen it is basically 25% for each. These will serve as our baseline metrics to compare against for our future models.

Hyper-parameter Tuning:

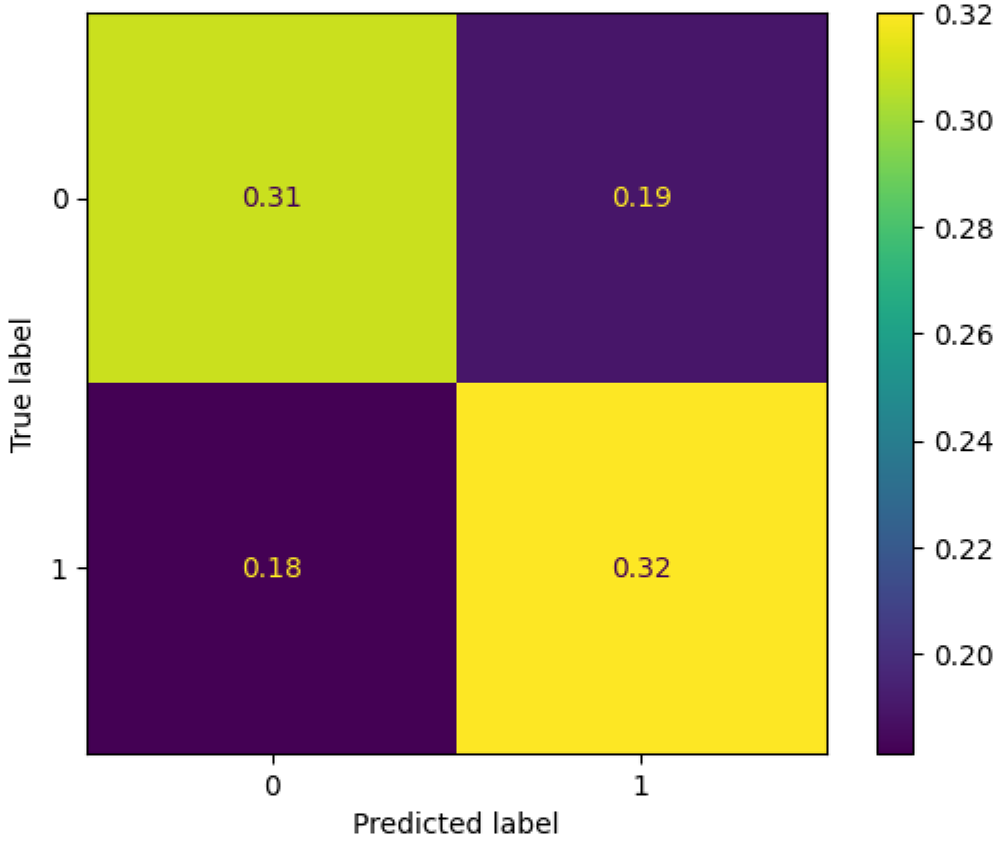
The random model did not require any tuning, as it was kept to be a uniform random value. However, the KNN and our neural network required some tuning. The KNN had three primary hyperparameters to tune: the number of neighbors, the weighting mechanism, and the distance metric. After running multiple tests we ended up with 518 neighbors, uniform or distance for the weights, and L1 for the distance metric. Increasing the number of neighbors helped, as it allowed for the decision boundary to be more smooth.

The neural network had a lot more hyperparameters to tune compared to the KNN. Probably the most important one was the structure of our neural network. Another thing we tuned was the activation functions. For our activation functions we ended up using LeakyRelu and at the end of the model was sigmoid, this was important as it mapped the final output value to be between zero and one. We decided to train at a fixed epoch limit of 20. Lastly, there was the optimizer to choose from and the hyperparameters associated with it. For our project, we were using PyTorch for our calculations, so we alternated from using SGD and AdamW. For SGD what helped training we increased the momentum to 0.9, which allowed the accuracy to increase quickly at the cost of stability.

Experimental Results

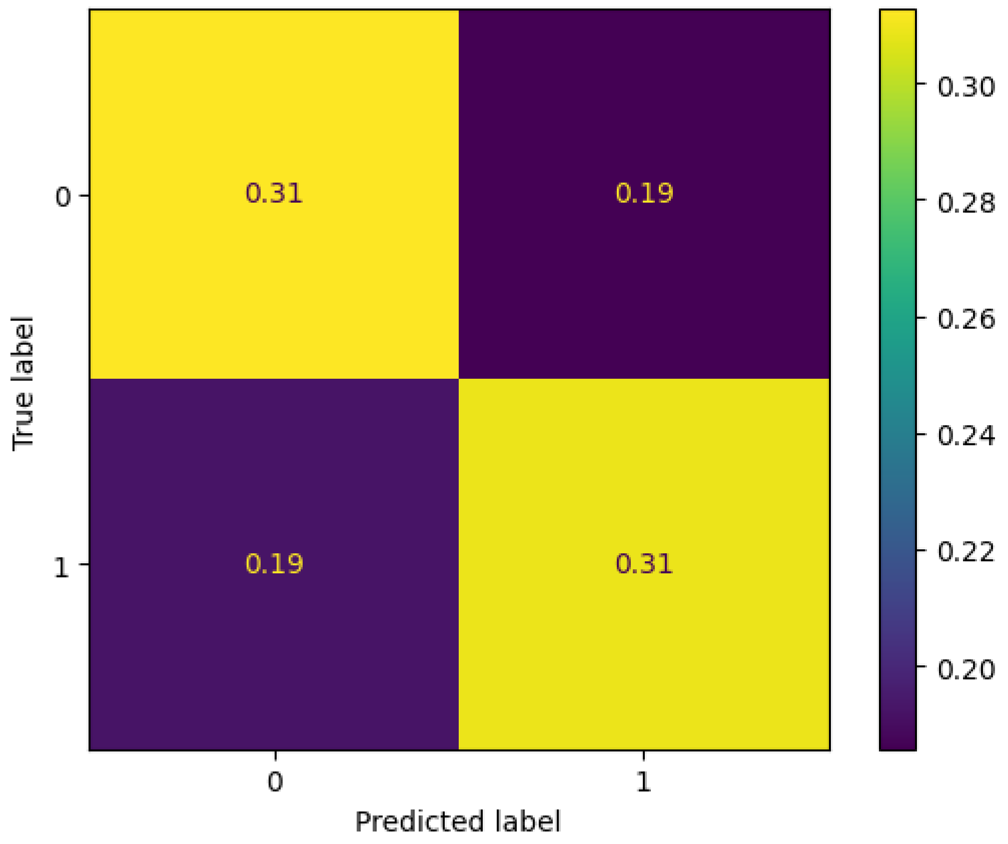
KNN (Uniform)

Class	Precision	Recall	F1-Score	Support
0 (Failure)	0.63	0.62	0.62	13344
1 (Success)	0.63	0.64	0.63	13427
Accuracy			0.63	26771
Macro Average	0.63	0.63	0.63	26771
Weighted Average	0.63	0.63	0.63	26771



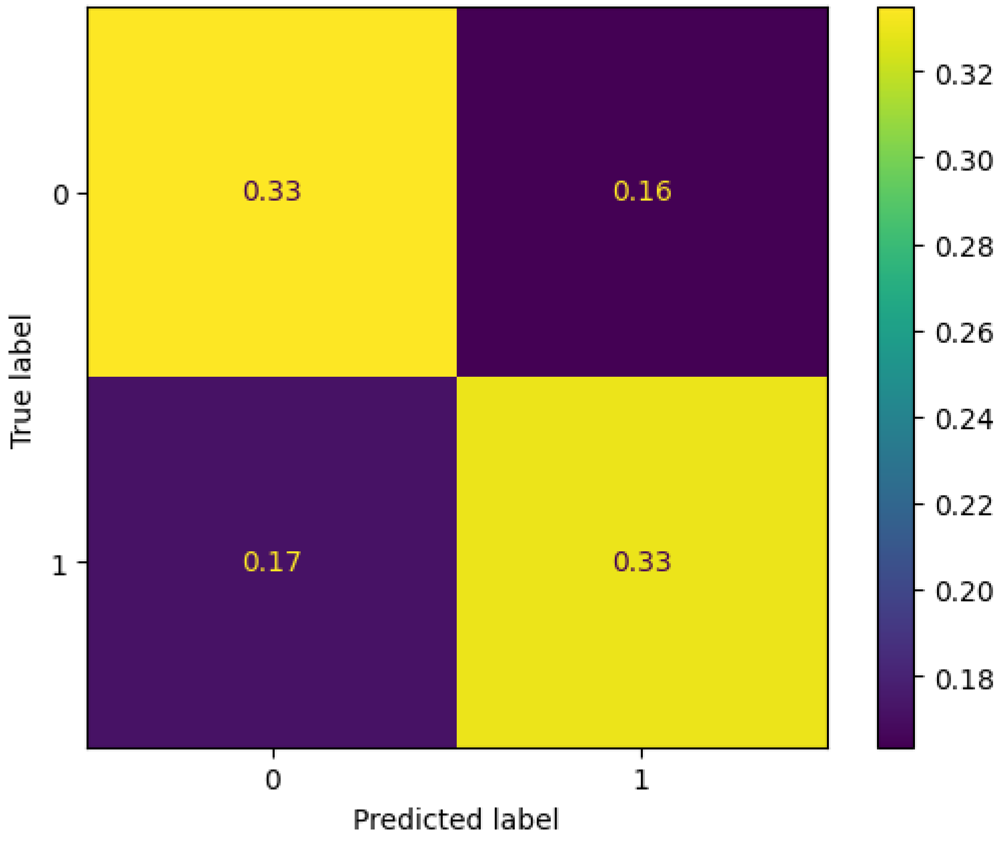
KNN (Distance)

Class	Precision	Recall	F1-Score	Support
0 (Failure)	0.62	0.63	0.62	13344
1 (Success)	0.62	0.62	0.62	13427
Accuracy			0.62	26771
Macro Average	0.62	0.62	0.62	26771
Weighted Average	0.62	0.62	0.62	26771



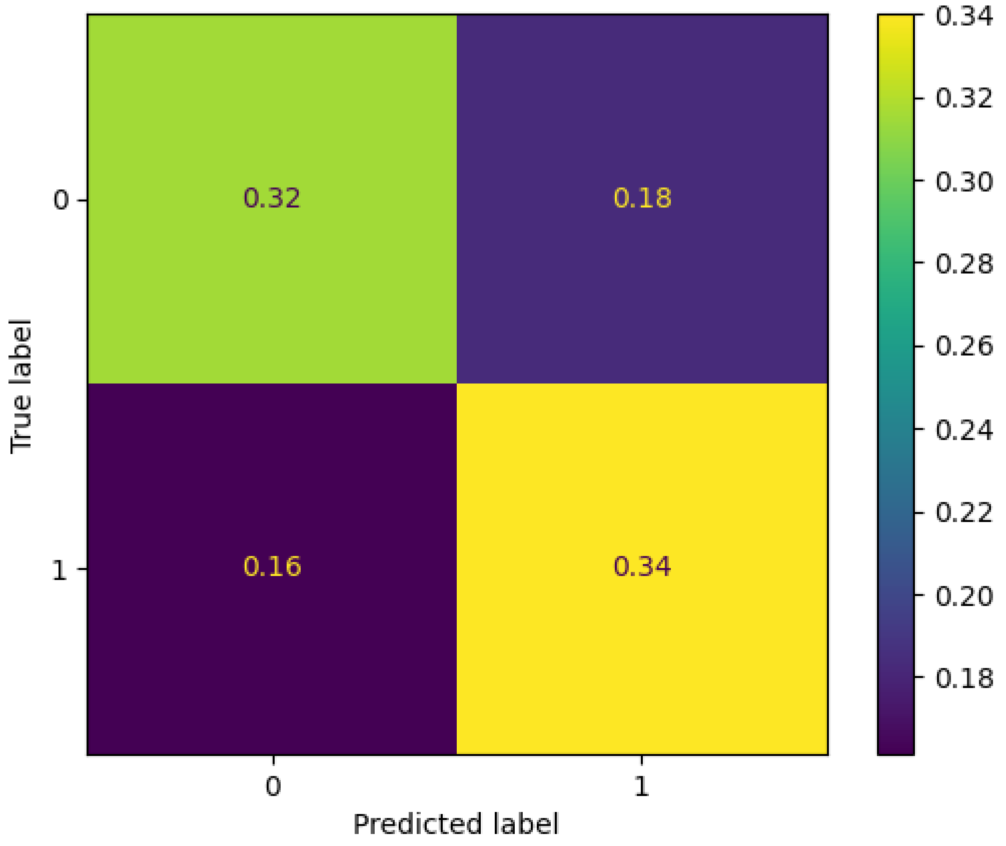
Neural Network (Simple)

Class	Precision	Recall	F1-Score	Support
0 (Failure)	0.66	0.67	0.67	13344
1 (Success)	0.67	0.66	0.66	13427
Accuracy			0.66	26771
Macro Average	0.66	0.66	0.66	26771
Weighted Average	0.66	0.66	0.66	26771



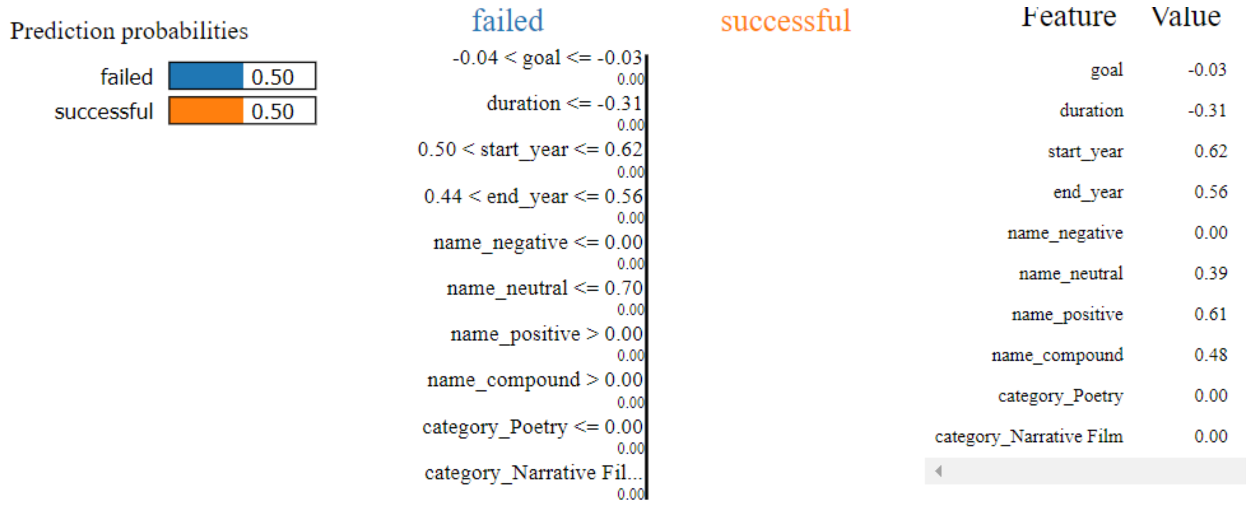
Neural Network (Complex)

Class	Precision	Recall	F1-Score	Support
0 (Failure)	0.66	0.63	0.65	13344
1 (Success)	0.65	0.68	0.66	13427
Accuracy			0.66	26771
Macro Average	0.66	0.66	0.66	26771
Weighted Average	0.66	0.66	0.66	26771



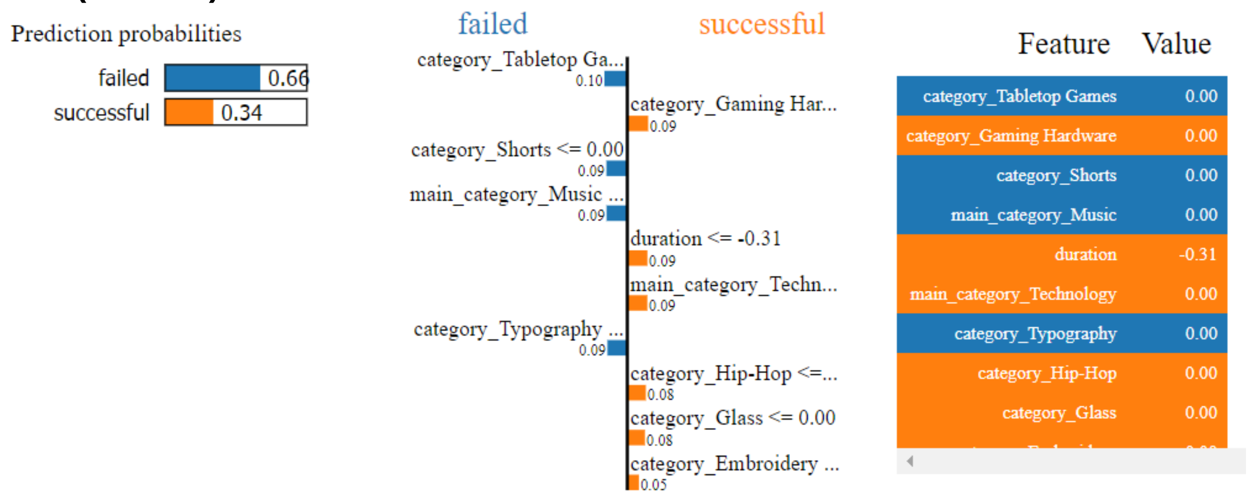
Lime for Failure

Baseline



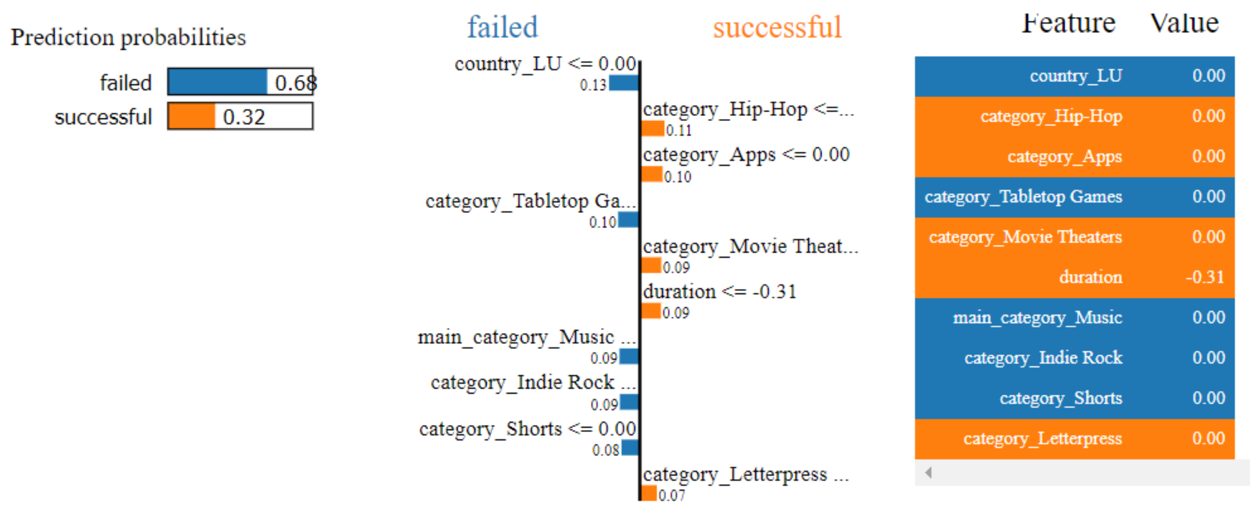
Feature	Value
goal	-0.03
duration	-0.31
start_year	0.62
end_year	0.56
name_negative	0.00
name_neutral	0.39
name_positive	0.61
name_compound	0.48
category_Poetry	0.00
category_Narrative Film	0.00

KNN (Uniform)



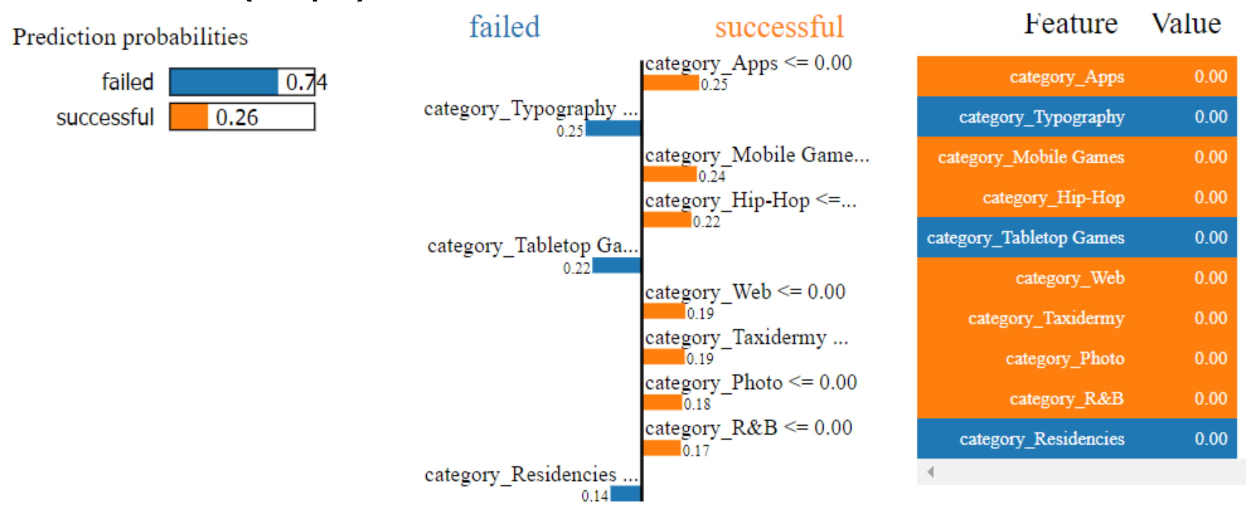
Feature	Value
category_Tabletop Games	0.00
category_Gaming Hardware	0.00
category_Shorts	0.00
main_category_Music	0.00
duration	-0.31
main_category_Technology	0.00
category_Typography	0.00
category_Hip-Hop	0.00
category_Glass	0.00

KNN (Distance)



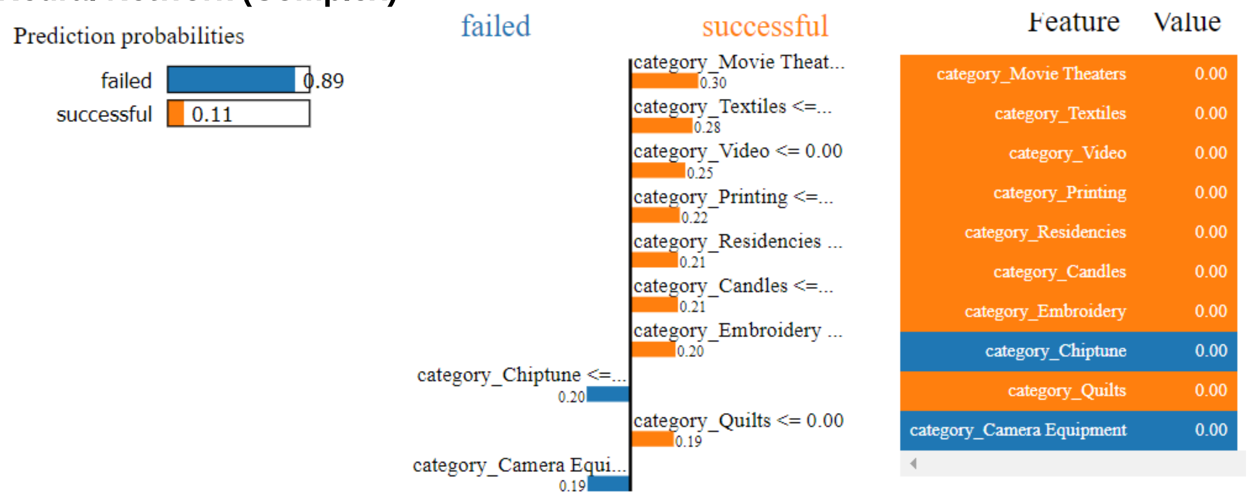
Feature	Value
country_LU	0.00
category_Hip-Hop	0.00
category_Apps	0.00
category_Tabletop Games	0.00
category_Movie Theaters	0.00
duration	-0.31
main_category_Music	0.00
category_Indie Rock	0.00
category_Shorts	0.00
category_Letterpress	0.00

Neural Network (Simple)



Feature	Value
category_Apps	0.00
category_Typography	0.00
category_Mobile Games	0.00
category_Hip-Hop	0.00
category_Tabletop Games	0.00
category_Web	0.00
category_Taxidermy	0.00
category_Photo	0.00
category_R&B	0.00
category_Residencies	0.00

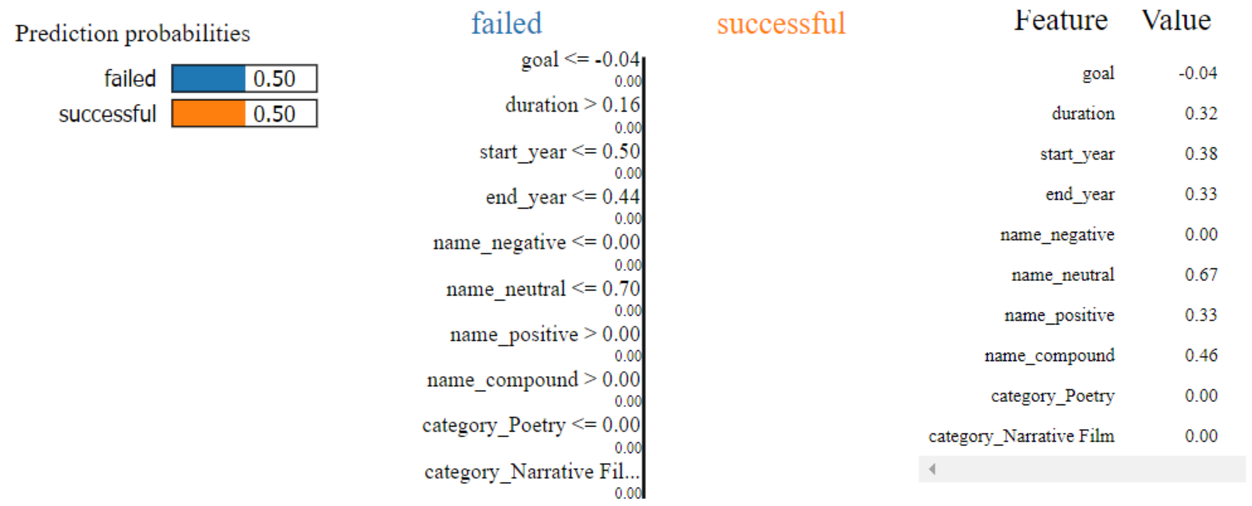
Neural Network (Complex)



Feature	Value
category_Movie Theaters	0.00
category_Textiles	0.00
category_Video	0.00
category_Printing	0.00
category_Residencies	0.00
category_Candles	0.00
category_Embroidery	0.00
category_Chiptune	0.00
category_Quilts	0.00
category_Camera Equipment	0.00

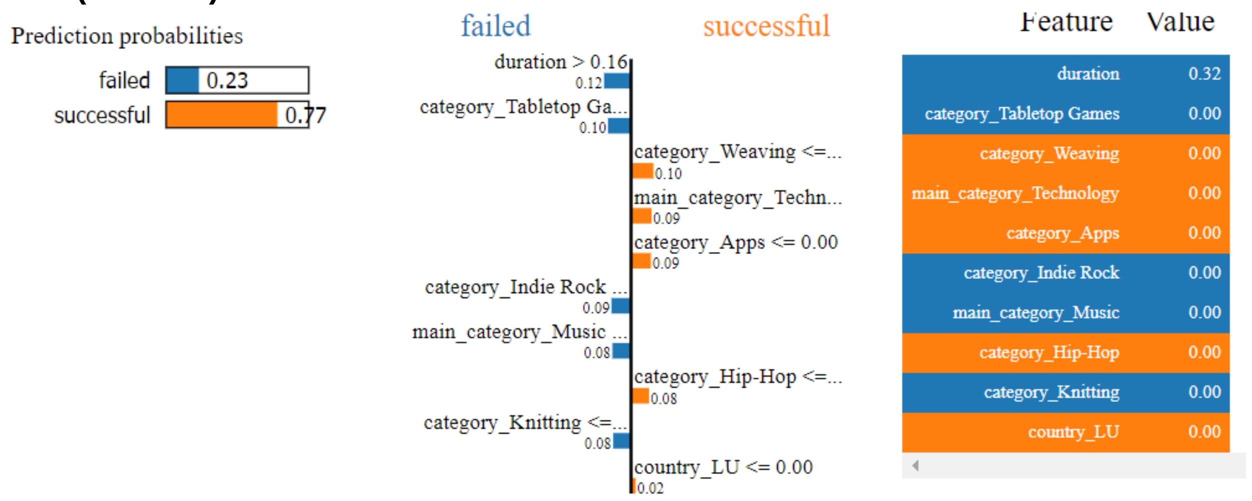
Lime for Success

Baseline



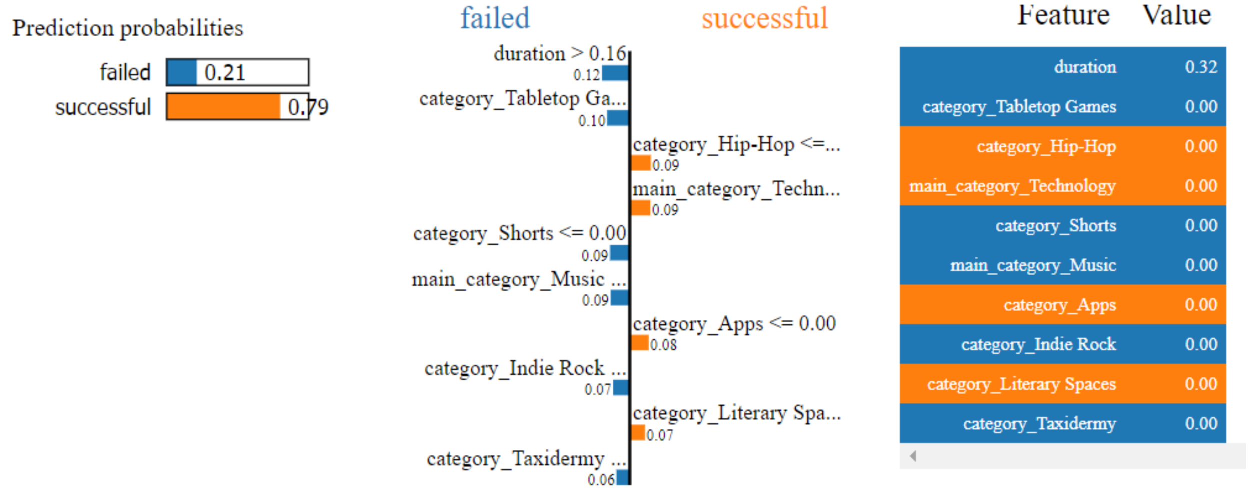
Feature	Value
goal	-0.04
duration	0.32
start_year	0.38
end_year	0.33
name_negative	0.00
name_neutral	0.67
name_positive	0.33
name_compound	0.46
category_Poetry	0.00
category_Narrative Film	0.00

KNN (Uniform)



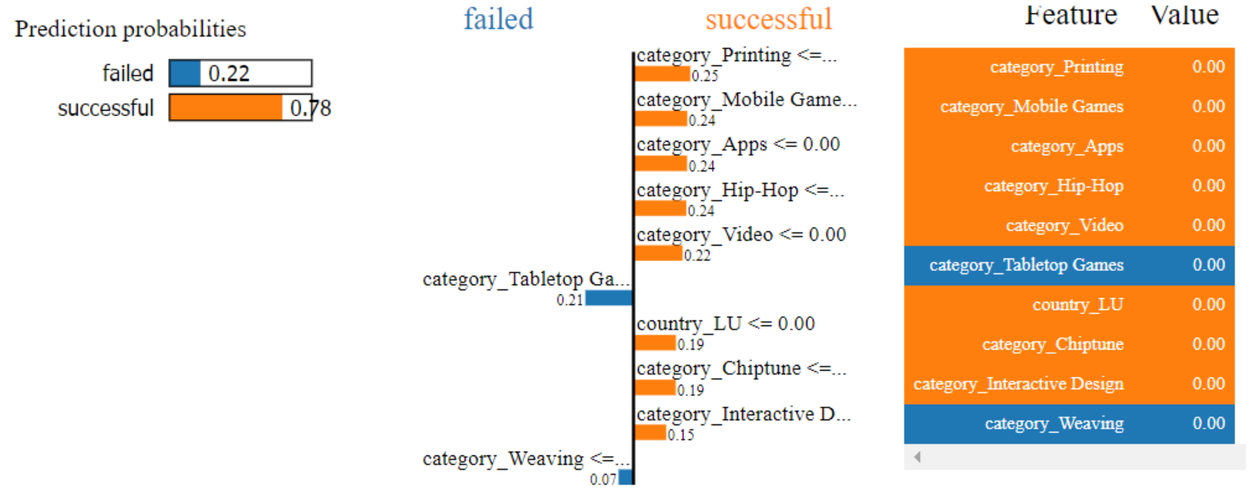
Feature	Value
duration	0.32
category_Tabletop Games	0.00
category_Weaving	0.00
main_category_Technology	0.00
category_Apps	0.00
category_Indie Rock	0.00
main_category_Music	0.00
category_Hip-Hop	0.00
category_Knitting	0.00
country_LU	0.00

KNN (Distance)



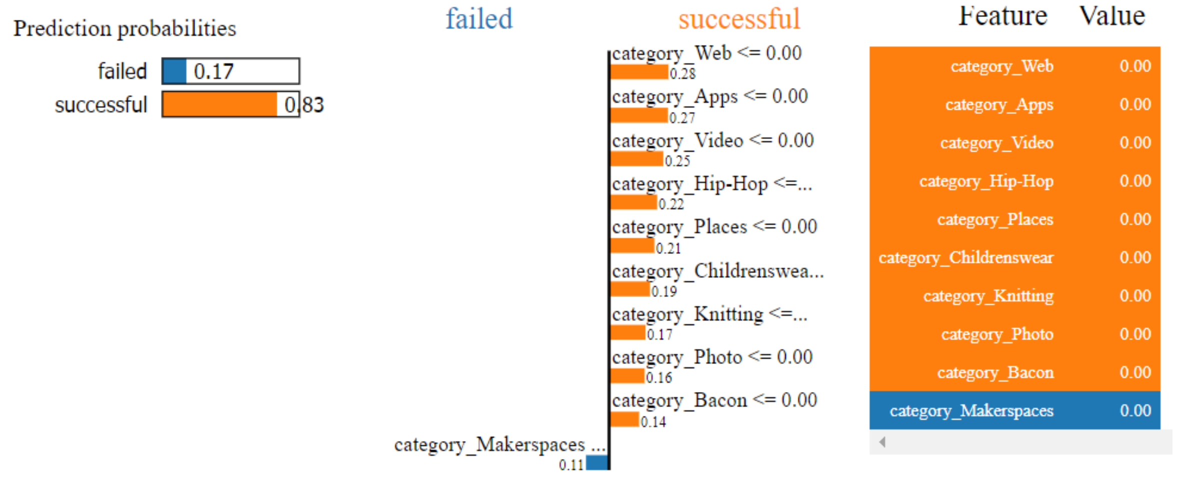
Feature	Value
duration	0.32
category_Tabletop Games	0.00
category_Hip-Hop	0.00
main_category_Technology	0.00
category_Shorts	0.00
main_category_Music	0.00
category_Apps	0.00
category_Indie Rock	0.00
category_Literary Spaces	0.00
category_Taxidermy	0.00

Neural Network (Simple)



Feature	Value
category_Printing	0.00
category_Mobile Games	0.00
category_Apps	0.00
category_Hip-Hop	0.00
category_Video	0.00
category_Tabletop Games	0.00
country_LU	0.00
category_Chiptune	0.00
category_Interactive Design	0.00
category_Weaving	0.00

Neural Network (Complex)



Feature	Value
category_Web	0.00
category_Apps	0.00
category_Video	0.00
category_Hip-Hop	0.00
category_Places	0.00
category_Childrenswear	0.00
category_Knitting	0.00
category_Photo	0.00
category_Bacon	0.00
category_Makerspaces	0.00

Analysis

Summary of Metrics:

Metric	Accuracy	Macro F1 - Score
Baseline	0.5	0.5
KNN (Uniform)	0.63	0.63
KNN (Distance)	0.62	0.62
Neural Network (Simple)	0.66	0.66
Neural Network (Complex)	0.66	0.66

Looking at the results of our experiments, while not the highest metrics, it does show that there is something interesting happening. The results show that you can somewhat predict a Kickstarter result. What this means in a sense is that a Kickstarter is not truly independent of other Kickstarters. If it was truly independent, then the accuracies would have been closer to 50%, making it less predictable.

To better understand the results we ran our models through LIME, which is a tool that allows a better understanding of the models, as otherwise, it is hard to understand what is happening. The results of LIME showed that our models were most of the time relying on a category, specifically not being in one, when determining success or failure. Future work should be done in order to avoid this bias of strong dependency in categories, compared to other features like goal or duration, as belonging or not belonging in a category, should not be the major deciding factor for success or failure.

Conclusion and Future Work

All in all, we used the methods taught in this class to analyze Kickstarter data. First, we determined which features to use and then modified them by either one hot encoding or normalization on them. Then using these prepared features we split them into training, testing, and verification sets. Finally, using these sets we trained KNNs and neural networks. In our future work, we would like to look into eliminating more of the biases in our models in the hope that it would increase our metrics and become more fair. One way we could do this is by gathering more data. Another option would be to train a separate model for each of the biases. Also, what we would like to do in the future is try other models, as they could produce better metrics. Lastly, we could add more features to our data. In the end, we showed that there exists some factors that do not make each Kickstarter truly independent of each other.

References

Lecture Slides

KNNformal definition Source: https://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote02_kNN.html#:~:text=The%20k%2DNN%20algorithm&text=Formally%20Sx%20is%20defined,furthest%20point%20in%20Sx).

Neural Networks formal definition Source: https://towardsdatascience.com/how-to-define-a-neural-network-as-a-mathematical-function-f7b820cde3f#:~:text=A%20k%2Dlayer%20neural%20network,%E2%88%98f2%E2%88%98f1

Kickstarter Data:
https://www.kaggle.com/code/kromel/kickstarter-successful-vs-failed/input?select=ks-projects-201801.csv4

LIME: https://github.com/marcotcr/lime

Python
Matplotlib
Numpy
Jupyter
PyTorch
Scikit Learn
nltk

Code

Our code Repository: https://github.com/CollinScott37/ACG

For one that has outputs it will be in the Blackboard submission, this is done for security concerns on not leaking information.