Predicting Kickstarter Campaign Success

Grace Kang, Lucas Parker, Josh Portelance, Frazer Chambers, Steve Hof

***Abstract*—The use of crowdfunding websites like Kickstarter has grown exponentially over the last several years leading to the funding of several successful companies. However, very few of the campaigns on Kickstarter are successful in meeting their funding goals. In this report we propose a strategy for building a model that predicts the likelihood of success of Kickstarter campaigns.**

1. Introduction

Kickstarter (kickstarter.com) launched in 2009 and has since revolutionized crowdfunding in the modern age. The core idea of Kickstarter is simple. It is all about letting anyone pitch a project which could be anything from technology and engineering, to art and potato salad [1]. Anyone with internet access can see all pitches and projects and donate to a project if they wish to help realize the project to its full potential. The projects that are pitched usually only need funding to be able to begin the real work on and eventually finish the project. A platform that allows for financial backers of any level from anywhere is Kickstarters contribution.

Each project pitch has a funding goal with different reward tiers for backers who donate certain amounts. For example, donating $5 to a movie might get your name thanked in the credits but donating $25 will get a you a copy of the movie when its finished. But not just anyone can make a fifty thousand dollar potato salad. All pitches have a time limit to get funded - and projects that do not raise their minimum funding goal by a specified deadline (30 to 60 days from start) will not receive any funding. Backers are only charged on projects they donate to that successfully reach their funding goal.

It would be of value for a project leader to know the percent chance that their prospective campaign will succeed so they can:

1) Know if it's worth their time and resources to pursue the campaign
2) Adjust the pitch to make it more likely to succeed. An incentive for our model to be human interpretable is so that we can find heuristics that leaders can follow to increase their success chance. A non-human interpretable model can still be of value facilitating item 1, and also for the purpose of choosing the best of multiple variants of a pitch.

Knowing the success chance of a particular campaign can also be of value to a potential pledger so they can make a more informed decision. The goal of our project was to accurately predict the success of Kickstarter campaigns. To do this, we used data mining techniques.

2. Related Work

Multiple classification approaches have been taken to predict Kickstarter campaign success; however, the topic has yet to be explored with neural networks. In a study done last year by R. Downs and M. Ghauri, a predictive model was built out of campaign attributes that had a high correlation with campaign success. A variety of modelling techniques were used with J48 Classification Trees yielding the highest accuracy of 71.9% [2].

In a recent study K. Sawhney, C. Tran, and R. Tuason from Stanford University built a binary classifier to predict the success of a campaign which used language instead of campaign statistics over time. To do this they built a binary classifier which analyzed campaign content, linguistic features, and meta-information at the time the campaign was created. Through sufficient training they were able to achieve over 70% accuracy with their classifier [3].

Another study done by V. Etter, M. Grossglauser, and P. Thiran from EPFL developed a model for predicting the success of a campaign using direct information such as money and pledges as well as social features such as Twitter activity over time. Using these predictors they were able to reach an accuracy of 76%; more specifically they found that their predictors based on money pledges over time were able to reach a high accuracy before combining them with the social features [4].

A project similar in nature to ours was done by N. Vr and S. Babu at the National Institute of Technology Calicut. They designed models to predict the success chance of movies based on certain features of the movie. To do this, they tried each of Linear Regression, Logistic Regression, and SVM. Their best results came from the Linear Regression, and they concluded that their prediction model was of nonzero value to society [5].

3. Dataset

We got our dataset from Data World [2]. It consists of approximately 20,000 points, each of which has multiple features and also the outcome of the campaign, success or failure. After removing the data points which had missing data for the features we were interested in there were a total of 18,736 points. Out of these campaigns, approximately 72.14% of them failed, and 27.86% succeeded.

Some of the attributes of the dataset that we used are shown in Table 1.

TABLE 1. ATTRIBUTES USED

Attribute	Format
Communication with campaign manager	boolean
Country of origin	categorical
Currency	categorical
Staff pick	boolean
Exchange rate with USD	float
Category	categorical
Spotlight	boolean
Duration (number of days campaigning)	int
Campaign Blurb	text
Campaign Goal	float
Number of Backers *	integer

It should be noted that Number of Backers was not included in our final algorithm. When it was included, the random forest achieved an accuracy of 92%.

Along with the attributes above, the dataset included all URLs corresponding to each campaign. This allowed us to extract two more textual components (TCs) of each campaign: the project description and the manager biography. Along with the blurb, which is a one sentence description of the project, we had three TCs.

As a large portion of a campaigns information is provided via its TCs, potential pledgers spend a significant amount of the attention they give to a campaign on its TCs. For this reason, we thought that incorporating aspects of the TCs into our models could increase our predictive power. We took two approaches to this. In the Conjuring Attributes section (5.1.1) we attempted to quantify the TCs on various dimensions, condensing each TC into values interpretable as attributes for the other models to use. In the Predicting Campaign Success from Blurb section (5.1.2) we used various algorithms to predict the probability of success of a campaign based on its blurb alone. The prediction from this model was then used as an attribute interpretable by the other models.

This section outlined the attributes of campaigns we had available for our models. Note that when implementing each model, we chose a subset of the available attributes that was optimal for the current task.

4. Project

Since Kickstarter follows an all or nothing funding model, backers and campaign creators generally care about whether or not a campaign is successful or not. If a campaign is unsuccessful, none of the backers will contribute to the project and the project will not be successful. We decided to first start by building predictive models that would simply predict whether or not a campaign will be successful. Then, building off of our findings we took it one step further by creating models that provide probability of success or failure of campaigns.

One of the previous studies (see Related Work) selected attributes that individually had a high correlation to campaign success as input for their predictive model. We decided to also include attributes which dont individually have a high correlation with campaign success. This decision was inspired by the observation of unpleasant tasting individual cake ingredients synthesizing into the ultimate orally ingestible hedonic experience. We hypothesized that Exchange rate with USD would be the flour of our Kickstarter cake.

Using this approach, we explored several different algorithms and combinations of features, compared their accuracies, and then combined the best models to create a single predictive model.

5. Completed Work

We took multiple approaches to predicting campaign success using the non-textual features of the dataset. These include a Decision Tree, a Neural Network, a Support Vector Machine. We also used Natural Language Processing techniques to predict campaign success based on the blurb alone. Then, we used the output from these models as features to be used for the approach that appeared most promising.

While creating the decision tree and training the neural network, we noticed that the "Spotlight" feature had a high correlation with the success of the Kickstarter campaign. It seems that almost all of the successful campaigns were spotlighted while very few if any failed campaigns were. Because of this, our model would consistently get 100% accuracy on our test data. To fix this we removed the "Spotlight" feature from our training data which allowed us to start properly training our models.

5.1. Natural Language Processing

5.1.1. Conjuring Attributes. In order to represent each body of text as values interpretable by our prospective models, we wanted a metric that was: common to each TC, minimally related to the context, naturally quantifiable, and indicative of success probability. We settled on two different metrics to use, each of which produced an attribute in its name for each TC.

The first metric we decided to use was positivity, as relating to sentiment. Emotional states can be transferred to others via emotional contagion, and emotional contagion occurs via text-based computer-mediated communication [6]. Consequently, a person reading text with a positive sentiment will likely feel more positive than if they were reading text with a negative sentiment. We thought this was relevant because people are more likely to donate to a campaign when their positive feelings are evoked by an advertisement for said campaign [7]. This led us to our first hypothesis: Campaign success probability is positively correlated with the positivity of the campaigns TCs.

The second metric we decided to use was complexity, which we defined synonymously with reading difficulty. We

thought that the complexity of the text would be indicative of success chance because less complex text is, by definition, easier to understand. This should lead to the average potential pledger gaining a better understanding of the project. With a better understanding, they may be more confident pledging. Increased confidence may lead to a more significant pledge. With more significant pledges, the campaign will be more likely to succeed.

This led to our second hypothesis: Campaign success probability will be negatively correlated with the complexity of the campaigns TCs.

To test our hypotheses, we first evaluated each campaigns TCs on the two different metrics. To measure positivity, we used a third party application that took a body of text as input, and output a positivity score in between 0 and 100. To measure complexity, we used the Flesch reading ease metric. This is a ubiquitous reading ease scoring system that is calculated as a function of the words per sentence and syllables per word. It takes a body of text as input, and outputs a rating from 0 to 206.835, with a higher score indicating less complexity. After obtaining the the evaluation scores, we split the data into two groups: the successful campaigns, and the failed campaigns. In each group, we calculated each of the mean positivity and complexity rating on each TC. Figure 1 and 2 show the results.

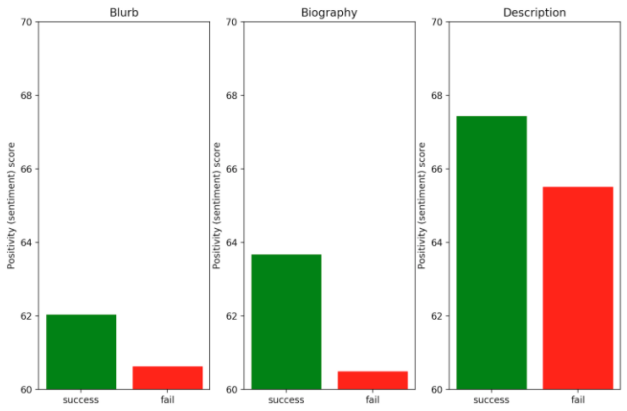

Figure 1. Mean positivity score for successful campaigns vs failed campaigns on each TC

The mean positivity rating of successful campaigns were greater than failed campaigns by 1.5, 3.6, and 2.1 for the blurb, biography, and description components, respectively. Because of these results, we accepted our first hypothesis.

The mean complexity rating of failed campaigns were slightly greater than that of failed campaigns, but not by a statistically significant margin. Because of this, we rejected our second hypothesis.

5.1.2. Predicting Campaign Success from Blurb. When potential pledgers are browsing campaigns on Kickstarter, the short blurb is what will attract the pledger to read more about the campaign. This makes the blurb an integral part of any Kickstarter campaign. Because of this, we deemed

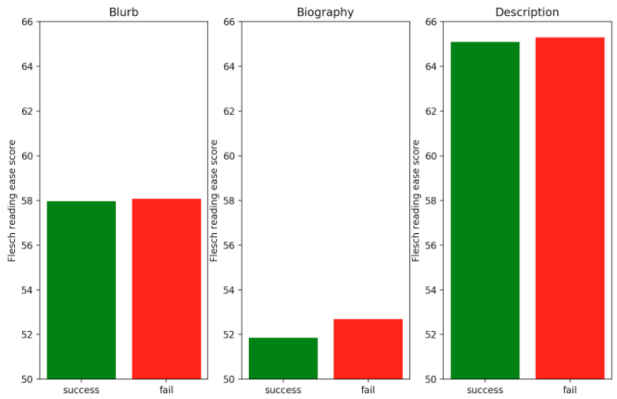

Figure 2. Mean Flesch reading ease score for successful campaigns vs failed campaigns on each TC

the blurb an important attribute to include in our predictive model.

We created two general models. One used a *Bag of Words* embedding as input and the other a *GloVe* embedding. [8] Both transform text data into a numerical form, making it interpretable as input to our learning models.

Bag of Words, the more straightforward of the two methods, first vectorizes the text corpus by determining all of the unique words in the entire data set and creating a feature column for each word. A blurb is then represented by a count of the number of times each of the words in the entire blurb corpus is used in that particular blurb. Before performing the *Bag of Words* vectorization we wanted to limit the dimensionality so we removed *stop words* ie. 'the', 'and' etc. as we considered their predictive value to be low. We also used Python's NLTK module to perform word 'stemming'. Word stemming is the process of reducing the number of words in our corpus by removing prefixes and suffixes. For example, instead of counting separate instances for 'worry', 'worrying', and 'worried', we reduced the latter two to 'worry', and gave it a count of three.

Since *Bag of Words* uses word counts in its predictions, we plotted histograms showing the most frequent words used in the blurbs over the entire dataset (Figure 3), and the most frequent words in blurbs where the campaigns were successful (Figure 4).

The words 'new', 'help', and 'world' are by far the most frequent across all campaigns, but after that there are some words more common in successful campaigns. *Bag of Words* models, by their nature, do not take into account the context of words in a sentence or paragraph but can be extremely successful none the less.

Sci-Kit Learn allows for the easy implementation of various algorithms once the text data has been cleaned, so we ran the blurbs through several of them. The accuracy results for the top three algorithms are shown in Table 2.

As can be seen from Table 2, the three algorithms performed better than the 70.8% baseline of predicting failure for every campaign, although barely. Concerned our top

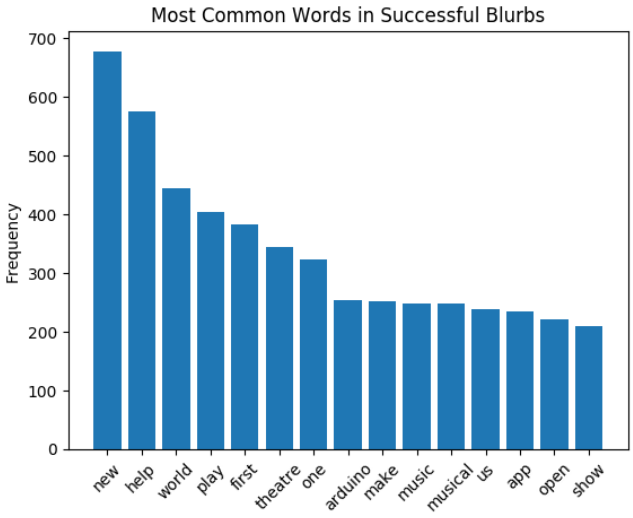

Figure 3. Most frequently used words in blurbs (stop words removed)

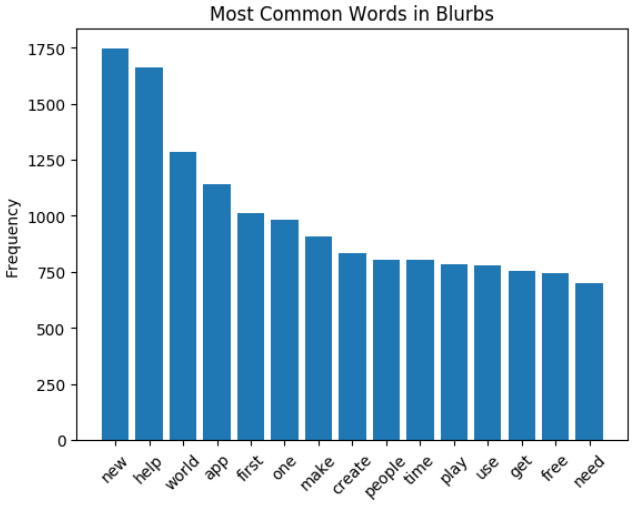

Figure 4. Most frequently used words in successful blurbs (stop words removed)

TABLE 2. ACCURACY FOR NLP ON THE BLURB FEATURE

Algorithm	Accuracy
Multinomial Naive Bayes	0.7401842
Bernoulli Naive Bayes (BNB)	0.7476975
Logistic Regression	0.7314590

algorithm (BNB) was simply choosing the majority class with a little bit of extra luck, we looked into the sort of blurbs Bernoulli NB was predicting to be successful and the results were quite interesting.

Figure 5 shows that our algorithm recognized that campaigns involving the performing arts were successful roughly 75% of the time and learned to recognize them. We found this result quite interesting as we're able to state quite confidently that those looking to raise money for a play, comedy show, etc. should take the time to create a Kickstarter campaign.

GloVe Embeddings: Unlike *Bag of Words* vectors, *GloVe*, short for Global Vectors for Word Representations, does take context into account. GloVe is an unsupervised learning algorithm for obtaining vector representations for words [8]. Using this algorithm, various sizes of pre-trained word vectors were created. For this project we chose the Common Crawl Corpus that contained 840B tokens, a 2.2 million word vocab, and vector feature spaces in 25, 50, 100, 200 and 300 dimensions.

In order to make use of these pre-trained word vectors, we chose to implement a Long Short Term (LSTM) Recurrent Neural Network (RNN) using Tensorflow [9] [10]. An RNN allows the model to remember words from earlier in a sequence of words in order to understand the context of the current word. This is fine for short sequences, but for longer sequences, such as a sentence or more of text, an RNN can carry forward too much information and run into one of two problems known as The *Vanishing* or *Exploding* gradient. We won't get into the details of them here, but an excellent explanation can be found at wildml.com [11].

In order to combat the gradient issues, we used an LSTM. In Chris Colah's excellent blog posthe states "LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!" [12].

In short, LSTMs use input and forget gates and learn activation rates for each in order to hold on to necessary information and let go of information deemed extraneous. After implementing the LSTM, it was immediately apparent that we had an overfitting issue. Our network was achieving close to 100% on the training data within a couple hundred epochs, but testing accuracy was leveling off at a maximum of roughly 70%.

In trying to overcome this overfitting we cycled through more than one hundred different combinations of hyperparameters for the number of LSTM nodes, number of hidden layers, learning rate, dropout layer keep probability, batch size, embedding dimension and epochs. In doing so we made a couple discoveries. Figure 6 shows a few of these combinations (all with two hidden layers) and Table 3 relates the hyperparameters to their short form in the legend.

TABLE 3. HYPERPARAMETER REFERENCES

Hyperparameter	Short form
LSTM Nodes per Layer	nLST-
Number of Hidden Layers	n_hidd-
Dropout Layer Keep Probability	dOut-
Unseen Testing Data	test
Training Data	train

As can be plainly seen from Figure 6, somewhere between 250 - 750 epochs the model's accuracy on the training data spikes toward 100%, while the testing data accuracy plateaus between 50% - 65%. This is a textbook case of overfitting. Our model is learning characteristics too specific

	blurb	Success	Prediction
7960	Parade Theatre Company and RHUL MTS present their Edinburgh Fringe 2016 production of Jason Robert Brown's The Last Five Years!	1	1
8497	Det nystartede vækstlagsteater Nørrebro Musicalteater's hårrejsende opsætning af horror-musicalen "Sweeney Todd"!	0	0.999995
7851	A new original solo show from an emerging performer and playwright set to premiere at the Hollywood Fringe Festival in June.	0	0.999994
6565	A fantastic feminist fairytale in four part harmony, premiering at the Edinburgh Festival Fringe this August 2014	1	0.999993
7374	Sh!t Theatre-supported emerging performance artist Eric Sigmundsson brings his debut solo-show to Edinburgh Fringe Festival 2015.	1	0.999983
6441	Inspired by the music of Kate Bush, we have devised an original piece of contemporary theatre to take to the Edinburgh Fringe 2016.	1	0.999982
7881	A graduate theatre company from Exeter Uni, performing 'Forever House' at Arts On The Move Festival and The Edinburgh Fringe 2015.	1	0.999962
6531	The play yet to be described as "A surefire Edinburgh Fringe Festival Cult Hit". Coming to the Underbelly, Edinburgh, 5th-30th August.	1	0.999961
5294	Voix de Ville is a pop-up imaginarium of neo-vaudeville, musical extravaganza, circus arts, comedy, and theatre in a tiny circus tent!	1	0.999957
728	ALLIE is a new dark comedy play which will premiere at the Edinburgh Festival Fringe 2015. Written and produced by Ruaraidh Murray.	1	0.999956

Figure 5. Blurbs the algorithm was most confident would lead to successful campaigns

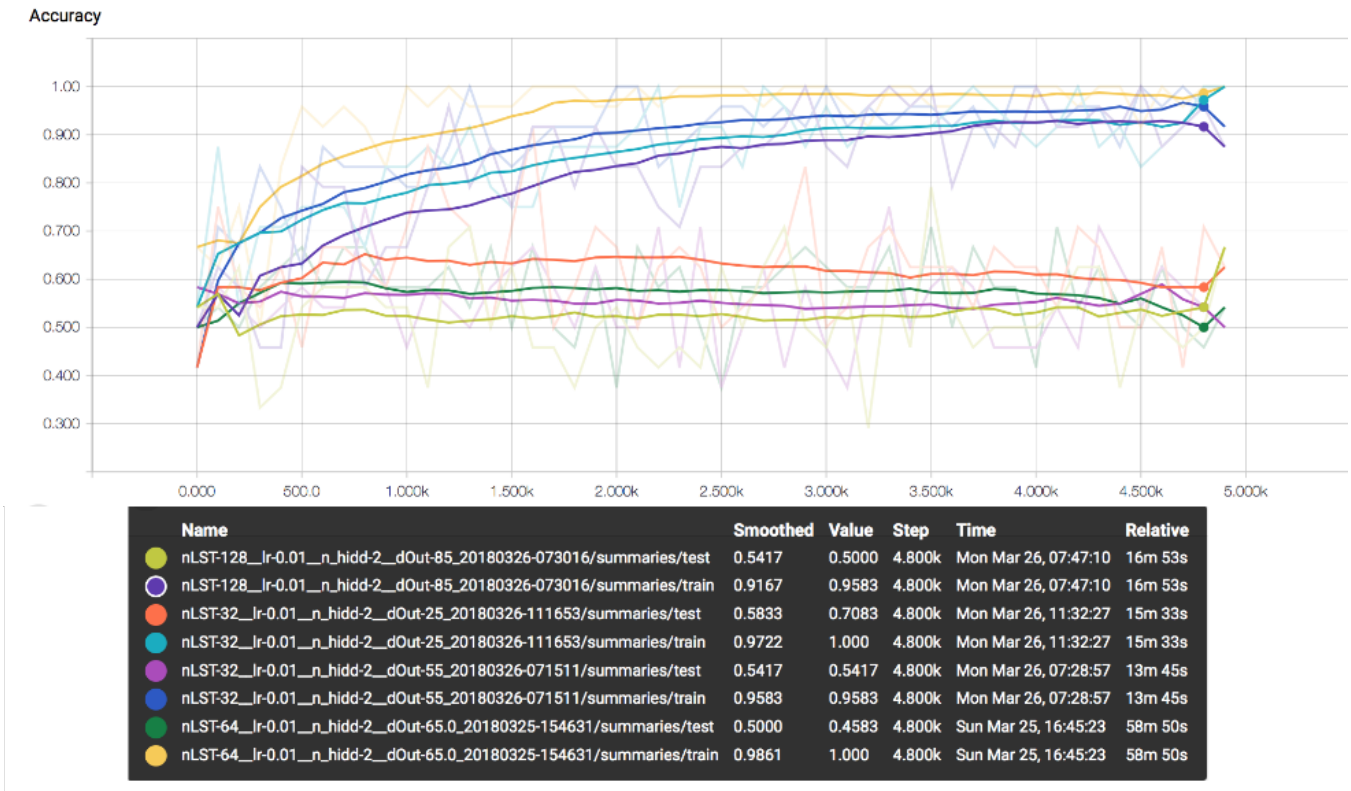

Figure 6. A plot showing the LSTM overfitting at a variety of hyper-parameter combinations

to the training data and is, therefore, unable to properly classify unseen testing data.

As mentioned previously, we tried a variety of different hyperparameter combinations, but found our results to either fall into the overfitting case shown in Figure 6 or the case seen in Figure 7, where we lowered the percentage of

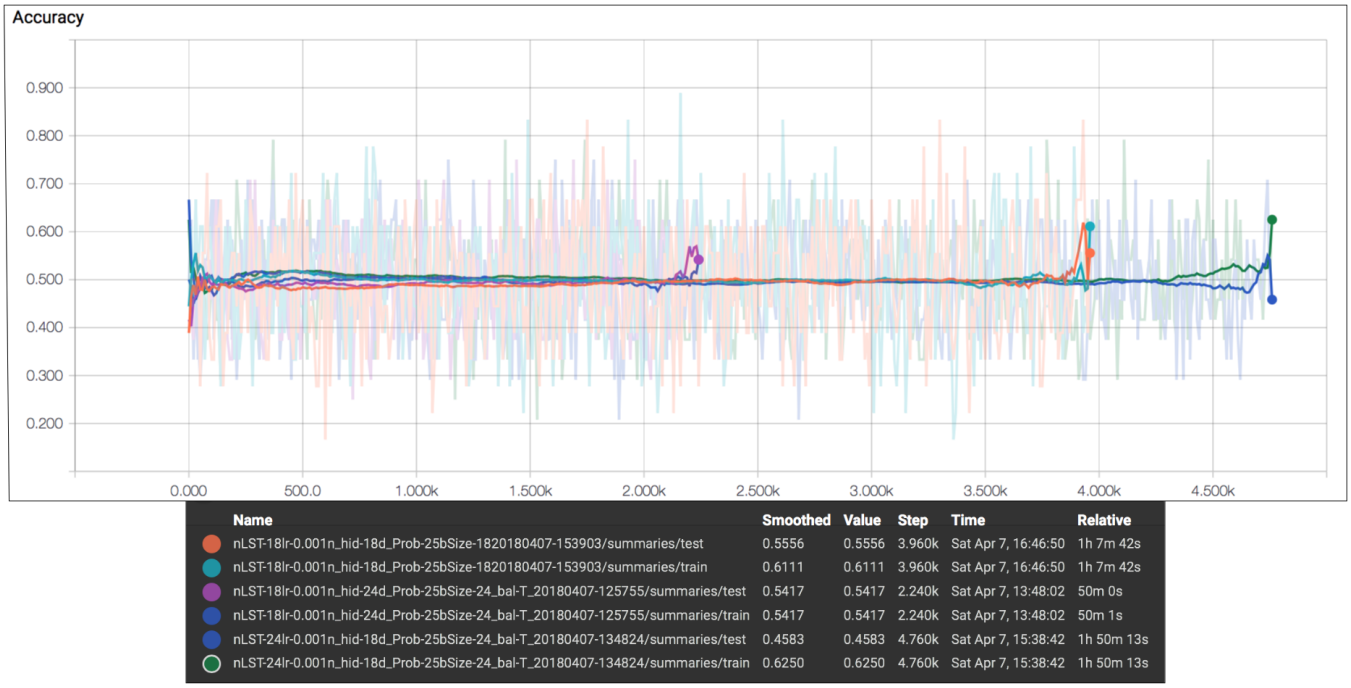

Figure 7. With more hidden layers and a lower dropout rate, the model doesn't learn at all

information that the dropout layer around our LSTM cells allowed to persist from layer to layer. In these cases we were able to prevent our model from overfitting, but in doing so also prevented it from learning at all.

5.2. Neural Network

Predicting Kickstarter campaign success is a binary classification problem so we built a feedforward NN to see how it would perform on our features. When building the feedforward NN, several approaches were used to try to achieve the best results. These included trying both full and batch gradient descent, adding and removing different features, normalizing the data, and applying one-hot encoding to categorical features. For each training iteration, the data was randomly split into 80% training data and 20% test data. We used L1Loss as our loss function, and RMSprop as our optimization algorithm

For each iteration of our NN, the data was randomly split into 80% training data and 20% test data, and we used L1Loss and RMSprop for our loss and optimization functions respectfully.

In order to decide if we needed to implement any hidden layers in our NN, we had to determine if our data was linearly separable. If the data was linearly separable, a NN without any hidden layers would be sufficient [13]. We went ahead and implemented the NN without any hidden layers and with one hidden layer. The NN with one hidden layer (Figure 9) yielded a higher accuracy on the testing data than the one without any hidden layers (Figure 8), evidently showing that the data is not linearly separable. After some trial and error with the number of hidden layers and the number of neurons per hidden layer we settled on an architecture that consisted of 3 hidden layers (Figure 8). The number of neurons per layer decreases in the following fashion: 60, 30, 15, 5, 1; where 60 is the number of features that are initially fed into the NN and 1 is our classification output.

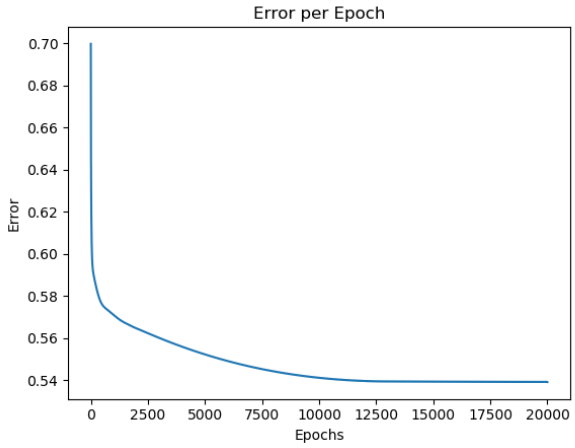

Figure 8. Error per training epoch of a NN with only a single linear layer

When using batch gradient descent the model quickly learned to simple predict failure for every case which would lead to the same accuracy if we were to guess false from the beginning. This occurred because as our data is quite unbalanced, there are many failed Kickstarter campaigns and only a few thousand successful ones; because of this

when using batch gradient descent it was possible to have several batches in a row which only had failed Kickstarter campaigns. To counter this we tried using full gradient descent to ensure the model was not being trained only on failed campaigns. By doing this we were able to train the model using the other approaches discussed in this section without having the unbalanced data negatively affect our results as much.

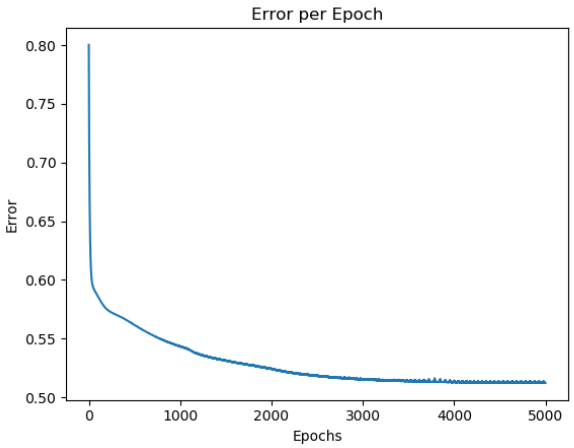

Figure 9. Error per training epoch of a NN with a single hidden layer

While trying to improve the accuracy of our predictions, we noticed that the dataset included more than just failed and successful projects. According to the State feature, there were also canceled, suspended, and live projects. Since these states are considered neither a success or fail, we thought that omitting them may result in better accuracy in our predictions as we are not trying to predict these states. However, this was not the case and eliminating the other states did not give a higher accuracy.

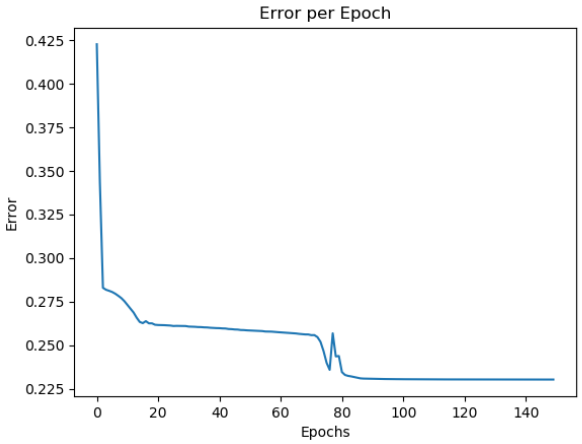

Figure 10. Error per training epoch of our final NN.

Using the final architecture that was previously described while also using full gradient descent and one-hot encoding for categorical features the NN was able to achieve an accuracy of 78.1%. This accuracy was obtained using only a subset of the features shown in Table 1, which included: communication with campaign manager, country of origin, currency, exchange rate with USD, category, and staff pick. The accuracy and error per training epoch of this final version of our NN are shown in Figure 10 and 11.

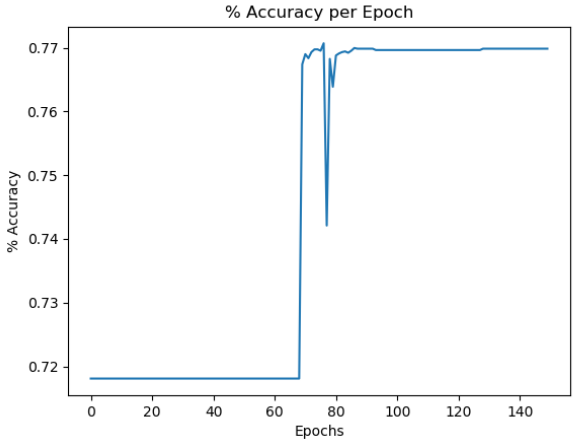

Figure 11. Percent accuracy per training epoch of our final NN

5.3. Support Vector Machine

A support vector machine was also created using Scikit-Learns SVC implementation. We chose to build a support vector machine because they are still effective when dealing with high dimensional datasets. The original dataset was initially split in half, 80% for training and 20% for testing. When building the support vector machine, four kernel functions were tested in attempt to achieve the highest accuracy during testing: RBF, polynomial, and linear functions from the SVC module, and also the LinearSVC module which is similar to SVCs linear kernel. Different C values (error penalty) were tested on the four different kernel functions as well. The results are shown in Table 4.

We found that the linear and RBF kernel functions performed the best on our dataset giving us a maximum accuracy of 77.0% on our test data. Increasing the C value improved the accuracy of the RBF and polynomial functions slightly but it did not improve the linear functions. When we attempted to use a C value greater than 100, the classifier was unable to train the data within a reasonable time duration. This indicates that outliers in our dataset have a large influence on the classifier and with our current computing resources, we are unable to find an optimal hyperplane that separates the data with the minimum margin.

5.4. Decision Tree

We made a decision tree classifier with the Scikit-Learn python library. Decision trees are one of the classification

TABLE 4. RESULTS OF SVM CLASSIFIER

Error Penalty	SVC (linear)	LinearSVC	SVC (rbf)	SVC (poly)
C = 1	77.0%	77.0%	76.2%	71.8%
C = 50	77.0%	76.9%	77.0%	76.4%
C = 100	77.0%	75.0%	77.0%	76.9%

techniques we explored as a predictive model for Kickstarter campaign success. We split our dataset randomly into 70% and 30% partitions. The former partition was used for fitting the tree and the latter was used afterwards for scoring the accuracy of predictions from the model. Our preliminary results suggest that the decision tree is quite an effective classifier with a classifier score of 0.88465.

Working with a decision tree also provided deeper insights about the attributes in our data set. Early on we were able to learn visual from looking at a decision tree that the spotlight feature automatically correlates 100% with campaign success. Including the spotlight feature in our models would be no different than including the outcome to be predicted in our input. In all our subsequent models we left out the spotlight feature.

Figure 12 shows a small view of a piece showing some leaves of the tree.

The tree is very large but given the large dataset this is expected. The dataset was further examined with this tree by applying labeling information to the nodes and attributes so we could see how the tree was coming to its decisions in a more human readable way. Pre-pruning the tree to down to 6 layers raised the accuracy scores from 88% to 90%, a larger view of the top and bottom of the tree is shown in Figure 13.

At this point (Figure 13) the tree used publicly available information from Kickstarter campaign web pages, conforming with our goal to create models that can predict campaign success based entirely off of information that would be available before starting a campaign. With such a model, given a potential Kickstarter campaign we can make suggestions as to how the campaign should change certain malleable attributes in order to increase their success chance.

We also built a random forest classifier. A random forest classier consists of multiple decision trees. Each tree is created with a random subset of the training data (with replacement). Splits that are chosen are no longer the ideal split among all features. Instead, the split that is chosen is the best split among a random subset of the features. Due to the overall randomness of many trees this model usually has less bias than a single decision tree. A forest of 27 trees each with depth 6 was found to have the best performance with an accuracy score of approximately 92%.

Figure 14 shows how increasing the number of days to the deadline is shown not to help chances of funding success. The solid blue line and the dotted red line represent the predictions of the decision tree and random forest, respectively. The random forest creates a more detailed line because, unlike the decision where a prediction score comes from one leaf, predictions from the forest are averaged from many leaves of many random trees.

Figure 15 shows how the tree and forest score the chance of success of a campaign based on the target goal. If the campaign asks for 100% of the original goal they have around a 50% chance to be funded. However, if they ask for less, their predicted success chance rises greatly.

Figure 16 is an example of another case where a campaign looks really good and is likely to be funded. This figure shows that the project could have asked for more funds, up to twice what they asked for and they would still have over a 75% chance to be funding.

With information like what is conveyed in Figures 14, 15, and 16, those who campaign on kickstarter could get an edge to help with funding, or in cases where funding already can be seen to be likely could consider such things as stretch goals or expanding the scope of their Kickstarter project and increasing the target goal.

6. Final Results

For our final model, we decided it would be most useful if the predictions with the information available before the a campaign launched. Thus, we removed "number of backers" from the dataset. To achieve our final result we decided to try combining several of the models we created together. To combine our models we first partitioned our dataset into two balanced halves. All of the models except for the combined one used one half of the data to train, and then make predictions on the testing half. We treated the predictions as attributes of the dataset, which was then used collectively as input for both a decision tree and a random forestcreate prediction for.

The decision tree and random forest classifiers used a 70/30 split of the half of the dataset with all model prediction scores. The 70% was used for training and 30% was used for evaluating the models accuracy. The results for each of the models before combining them together are shown in Table 5.

TABLE 5. FINAL ACCURACIES OF ALL OF THE INDIVIDUAL MODELS CREATED

Model	Accuracy
Decision Tree (without backers)	72.7%
Random Forest (without backers)	73.2%
Support Vector Machine	77.0%
Neural Network	78.1%
NLP Success from Blurb	73.2%

We started by only using predictions from one additional model at a time to see how they each affected the results. The accuracies when using the SVM, NN, and NLP model predictions are shown in Table 6.

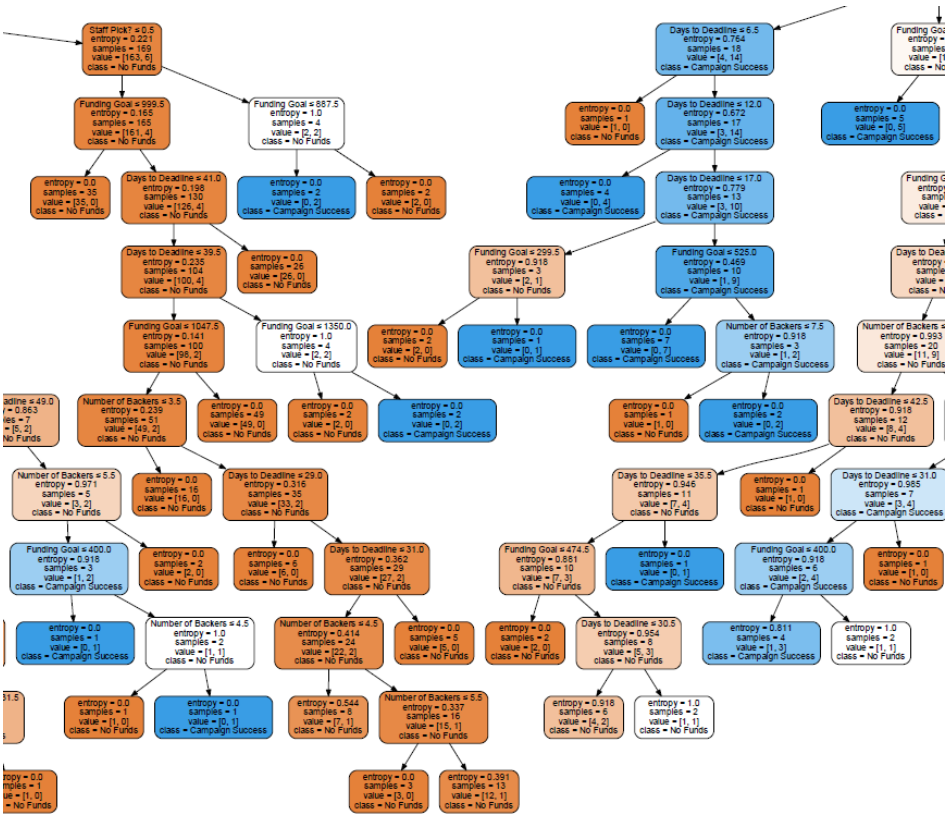

Figure 12. Decision Tree without pre-pruning

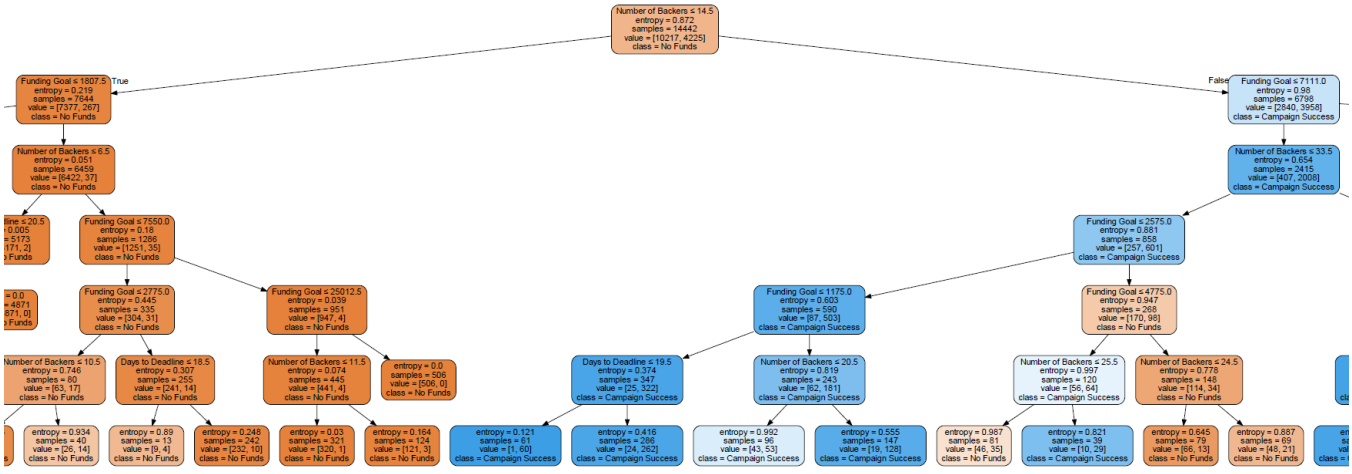

Figure 13. Labeled Decision Tree with pre-pruning (max depth of 6).

TABLE 6. DECISON TREE AND RANDOM FOREST ACCURACIES (WITH PREDICTIONS FROM INDIVIDUAL MODELS USED AS INPUT)

Prediction Model	Decision Tree Accuracy	Random Forest Accuracy
without any prediction features	72.7%	73.2%
with just the SVM predictions	77.3%	77.1%
with just the NN predictions	77.1%	77.5%
with just the Blurb predictions	74.9%	76.4%

Finally, we combined all of the models together to achieve our final accuracy of 77.9% with a decision tree,

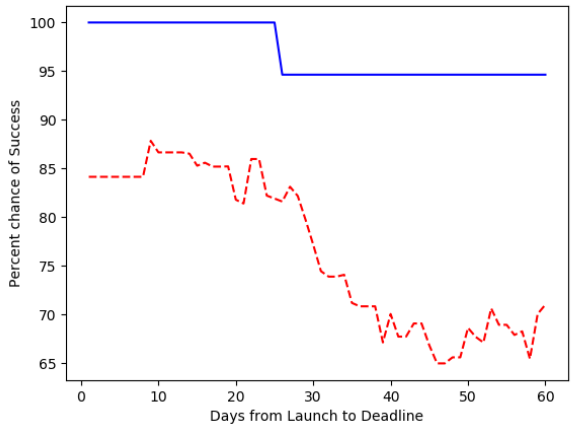

Figure 14. Predicted chance of success over ”days to deadline.”

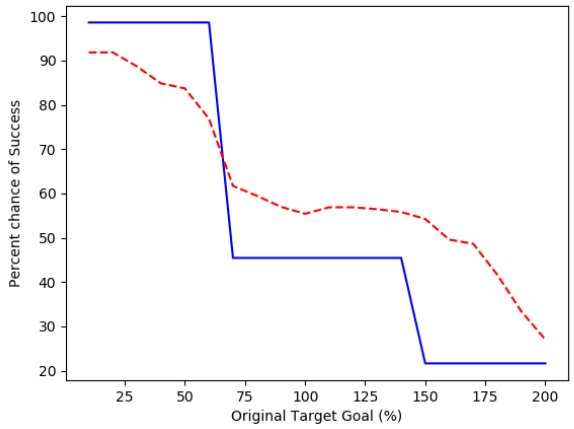

Figure 15. Predicted chance of success over ”Target Goal” (percent of original goal)

and 78.3% with a random forest.

While trying out different combinations of models we also tried using the predictions from the NN and the BNB classifier on the blurbs as inputs into the SVM. Throughout the process of trying different kernel types and values for the penalty we were able to obtain the best accuracies for each combination as shown in Table 7.

TABLE 7. ACCURACY OF EACH KERNEL TYPE WITH THE OPTIMUM PENALTY VALUE FOUND.

Kernel Type	Penalty Value	Accuracy
RBF	100	77.8
Poly	100	77.8
Linear	1	77.7
LinearSVC	10	78.3

As you can see in Table 7, the best accuracies we obtained by using the predictions from the NN and the

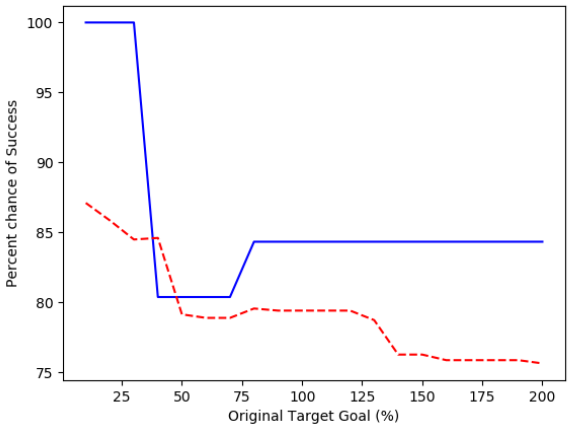

Figure 16. Predicted chance of success over ”Target Goal” (percent of original goal)

Blurb as inputs into the SVM was 78.3% using a LinearSVC kernel with a penalty of 10.

Next we tried using the original predictions from the SVM and the predictions from the BNB blurb analysis as input into the NN. Nothing was changed in the NN other then changing the number of neurons in the initial layer. With the addition of the two new features we were able to achieve an accuracy of 78.8%, with a much more stable error as shown in Figure 17.

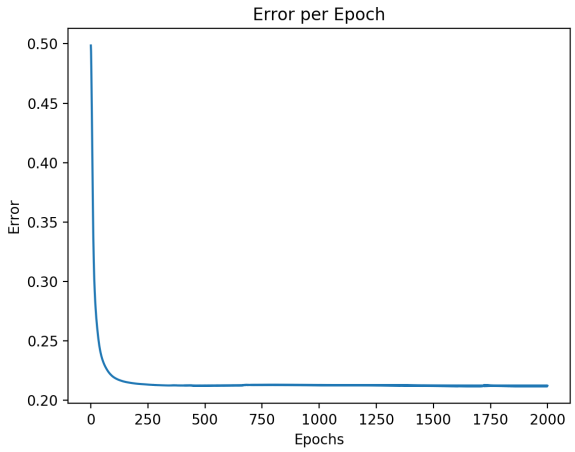

Figure 17. Error per training epoch of the NN with the additional features of the predictions from the SVM and Bernouilli Naive Bayes analysis

6.1. Conclusion

The best accuracy we were able to achieve was 78.8% using the predictions from the Bernoulli Naive Bayes classifier on the blurbs, NLP analysis and the SVM as input into our NN. The next best models were tied with 78.3%

accuracies which were SVM using NN and Blurb NPL analysis predictions as features, and a random forest using NN and BNB NLP analysis predictions as features.

7. Future Work

Thus far our LSTM neural network has not been a useful predictor. However, because we feel the ability to predict campaign success from the blurb alone to be vital to our project moving forward (past the due date for our SENG 474 class), we will continue to research and tweak our model in the hopes of it proving more reliable than the 73.2% achieved by our Bernouilli Naive Bayes Bag of Words model.

In the future we plan to work on trying to use other features either from both our dataset and output from new models. For example, with additional time we should be able to collect positivity scores for more our campaign samples and, therefore, be able to use that rating as input to our ensemble algorithms.

When we have our final model more defined, we plan to deploy it as a web application to evaluate Kickstarter campaigns in realtime and predict their success probability. Our application could also provide suggestions on changes to make in order to increase the probability of the campaign being successful.

8. Social Concerns

In the future, if our models are implemented into a web application to evaluate Kickstarter campaigns, we will have to take into account several social and ethical concerns.

A web based application that evaluates Kickstarter campaigns may be a useful resource for project creators who wish to improve a current campaign or increase its success rate. These predictions, however, could also influence project backers and lead to biased decisions on whether or not to fund a specific campaign. If a large portion of project backers were to use this application, a positive prediction for a specific campaign could result in more people funding it because people want to support projects that will succeed. This can result in an unfair advantage for the campaigns that are predicted to succeed.

In addition, we need to have policies and procedures in place to ensure our models do not accidentally favour or punish campaigns created by by people of a certain race, sexual identity or social class.

The possibility that our app could discourage what would have been successful Kickstarter projects from launching because of erroneously low predicted rates of success.

People that can afford our consulting will be more likely to succeed than people who cannot. This will infringe on the equal opportunity that our society strives to provide.

References

[1] Z.Brown. (2014) Potato salad. [Online]. Available: https://www.kickstarter.com/projects/zackdangerbrown/potato-salad

[2] R.Downs and M.Ghauri. Predicting kickstarter campaign success. [Online]. Available: http://racheldowns.co/portfolios/predicting-kickstarter-campaign-success/

[3] C. K.Sawhney and R.Tuason. (2016) Using language to predict kickstarter success. [Online]. Available: http://web.stanford.edu/~kartiks2/kickstarter.pdf

[4] M. V.Etter and P.Thiran. (2013) Launch or go home! [Online]. Available: https://infoscience.epfl.ch/record/189675/files/etter2013cosn.pdf

[5] N.Vr and S.Babu. Predicting movie success based on imdb data. [Online]. Available: https://www.researchgate.net/publication/282133920_Predicting_Movie_Success_Based_on_IMDB_Data

[6] A. Kramer, J. Guillory, and J. Hancock. Experimental evidence of massive-scale emotional contagion through social networks. [Online]. Available: http://www.pnas.org/content/111/24/8788

[7] T. Faseur and M. Geuens. ommunicating the right emotion to generate help for connected versus unconnected others. [Online]. Available: http://journals.sagepub.com/doi/abs/10.1177/0093650210368280#articleCitationDownloadContainer

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[10] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1999.

[11] D. Britz. Recurrent neural networks tutorial, part 3. [Online]. Available: http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-

[12] C. Colah. (2015, August) Understanding lstm networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13] J. Heaton. (2017, June) The number of hidden layers. [Online]. Available: http://www.heatonresearch.com/2017/06/01/hidden-layers.html

TABLE 8. DISTRIBUTIONS OF TASKS

Task	Team Member
Neural Network, Ensemble Models, Extra work on report	Josh
Neural Network, SVM, Ensemble Models, Oversaw Presentation	Grace
Decison Trees, Ensemble Models, Extra work on presentation	Frazer
Positivity and Complexity of text data, Extra work on report	Lucas
NLP on Blurb Data, Ensemble Models, Oversaw and compiled reports	Steve