It's All Funds & Games

Predicting Kickstarter Success

Mark Giannini
University of Notre Dame
mgianni1@nd.edu

Patrick Tinsley
University of Notre Dame
ptinsley@nd.edu

Brian Tunnell
University of Notre Dame
btunnell@nd.edu

1. Introduction

For our semester project, we decided to test our ability to predict whether a given Kickstarter campaign would be successful or not. In order to be deemed a success, the campaign in question needs to meet or exceed the funding goal proposed by the initial author by a predefined deadline; anyone can contribute to the project as long as the campaign is still active. Each instance in our data set has a unique project ID and twenty features; these include project name, financial goal, country of origin, category, project launch date, and project deadline. The target variable is labeled SuccessfulBool, and is coded as True for a success and False for a failure.

2. Related Work

Our first data set originally came from Kaggle, a competition-based data science website that houses thousands of public data sets. Consequently, there are several kernels or project templates that address this feat of predicting Kickstarter campaign success. However, since the service is innately challenge-based, most submissions are not visible to users.

Nevertheless, we did find the work of many members of the online data science community available for perusal and inspiration. One such project dealt most directly with spatial and geographical features of the campaigns, such as, country, state, and county; that work can be reviewed here. Another project focused primarily on the categories of the campaigns, such as music, theatre, tech products, and art; that work can be reviewed here. In the latter attempt, the authors found accuracy rates of .63, .68, and .65 for kNN, random forest, and logistic regression, respectively. For our project, we hope to supply a model that surpasses these accuracy benchmarks with the ability to generalize to future projects.

3. Problem Definition

The problem of determining the success of Kickstarter campaigns is inherently a classification problem; a campaign either succeeds or fails. In the context of our data set, this is represented in the SuccessfulBool field, which contains only either False or True for failures and successes, respectively. Given the binary nature of the target variable, logistic regression techniques will be used throughout the model construction and evaluation process.

, .

http://dx.doi.org/10.1145/

4. Proposed Methodology

The direction and methodology of our project included using several models provided by the scikit-learn Python library. After securing, cleaning, and splitting our data set, we fit six models with the following techniques: decision tree, random forest, logistic regression, naive Bayes, k-nearest neighbors, and adaptive boosting. Each of these models was fed through GridSearchCV (another model selection function provided by sklearn) to optimize each estimators hyperparameters; this exhaustive search over pre-specified parameter sets ensures that we ultimately select the highest-performing parameter combination for each model.

Furthermore, after fitting these first six models, we constructed a seventh model using the TPOT library, an automated machine learning tool provided by EpistasisLab that optimizes machine learning pipelines using genetic programming. TPOTs AutoML technique resembles the GridSearchCV technique mentioned earlier, but expands to include both hyper-parameters as well as estimators. Given the vast model-parameter space, we knowingly expected this process to be expensive in terms of time and computing power.

In terms of model evaluation, we recorded the following metrics for each estimator: accuracy, precision, recall, F1 score, and area under the receiver operating curve (AUC).

5. Data & Experiments

5.1 Data Set

Originally, we planned on crawling the Kickstarter website itself to collect data. However, upon browsing Kaggle, the data science competition hub mentioned earlier, we found a data set that contained data for more than 100,000 campaigns. Beyond a unique project ID, each row in the set contained fourteen features, including: project name, project description, project keywords, financial goal, project launch date, project deadline as well as the number of backers supporting the project. We applied the proposed methodology as described above on this data from Kaggle, and we found sublime results, as discussed in our midterm milestone report. We reported an accuracy and AUC both greater than .9.

Though initially very exciting, these exceptional results encouraged the group to take a second look at our data, where we found a problem with data leakage. The inclusion of the backer_count variable and state_changed_at variable (which documents when the project is marked successful or failed) gave away information that made campaign classification rather effortless. For example, if the

state_changed_at date occurs before deadline date, the estimator can presume success with an accuracy of .9; the remaining percent are projects marked failures by the authors before the campaign runs its full course. In this way, future information related to the target success category is leaked into the past. Additionally, at the time of project inception, these values mentioned above are unknown, so the use of that information in any model immediately renders it irrelevant in the context of the real world.

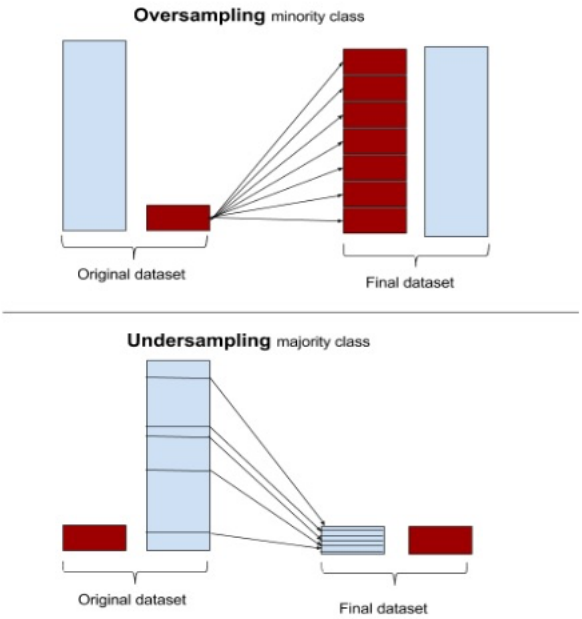
After removing these two features, we ran the newer data through the proposed methodology, and reported an accuracy and AUC around .8. These results were more believable, but we remained unsatisfied with our data and results. Firstly, we found the most important features to be uninteresting, such as expected duration, launch date hour, and project goal. One of the main objectives of this project was to find interesting, actionable features of promising campaigns. One such feature was the project category, which we hoped to ascertain by clustering text data from the project name, description, and keyword features. However, our two k-means clustering algorithms, which used a TfidfVectorizer and a HashingVectorizer, produced unreliable results, clustering 60,000 of the 100,000 projects in one category. Ultimately, we decided to explore an entirely new data set, provided by the data.world website.

Our final data set contained data for 18,738 projects with twenty features. Satisfying our qualms listed above, this set contained more interesting variables, such as:

- name_len_clean': an integer describing the length of the project name stripped of punctuation and emojis
- 'blurb_len_clean: an integer describing the length of the project description stripped of punctuation and emojis
- disable_communication: a Boolean describing whether or not the campaign author allows communication with contributors
- 'launched_at_weekday': a categorical variable describing the day of the week on which the project was launched (deadline_weekday is the deadline date equivalent)
- staff_pick: a Boolean describing whether or not the campaign is initially selected by Kickstarter staff and recommendation engines as an up-and-coming project
- category: a categorical variable describing which of the 25 categories the campaign belongs to
- 'launch_to_deadline_days': an integer describing the expected duration between launch date and deadline date

After finalizing our data set, we immediately dummy coded the categorical variables using the pandas get_dummies function, which increased our feature count to 217. The next step was dealing with class imbalance. Regrettably, most of proposed Kickstarter campaigns end up failing. In the case of our data set, 13,518 projects failed while 5220 succeeded. In order to remedy this imbalance, we under-sampled our data set. We chose under-sampling instead of over-sampling because, in our case, the latter would draw duplicate projects from the much larger negative class. In some cases, this would make sense, such as the iris data set, where some instances of flowers may have the exact same dimensions to one decimal point. However, no Kickstarter campaign is exactly the same as another, so creating facsimile projects would be inappropriate.

As can be seen in the image below, we reduced the number of projects used from the majority class (unsuccessful funding) in order to more precisely balance the count of successful and unsuccessful funding results. The resulting data set contained 5,220*2 = 10,440 projects.



5.2 Experimental Settings

To evaluate our models predictive power, we split our data set into two sets. The first partition is the training set, which is used to build and train the models. The second partition is the testing set, which is used to validate and critique the model. We split the data 75-25% respectively, resulting in a training set with 7,830 rows, and a testing set with 2,610 rows. By withholding a subset of the full data set, we have the power to test our final model on unseen data, which can be used to evaluate estimator performance; this technique also helps to avoid over-fitting and produces a more generalized model capable of predicting the success of future Kickstarter projects. We generated this split by using the train_test_split function from sklearn.model_selection after the under-sampling process mentioned above.

5.3 Evaluation Results

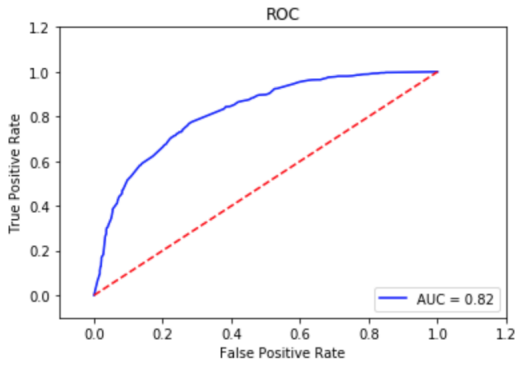
5.3.1 Decision Tree

Initially, we decided to fit a decision tree model to our training data. Using GridSearchCV provided by scikit-learn, we found that the optimal decision tree consisted of a Gini index criterion and max depth of nine. Decision tree models are effective for showing statistically important predictors, and the classification tree is easy to interpret and explain; the full tree can be seen by running the code or by clicking here. The recursive, top-down procedure selects features that result in splits that minimize uncertainty as much as possible (in this case calculated using the Gini index). Feature importance is calculated using the (normalized) total reduction of the criterion brought by that feature. For our tree, the most important features are as follows:

Feature	Importance
goal	.3379
staff_pick	.2307
category_Web	.1225
category_Software	.0747
name_len_clean	.0617

This is not entirely unexpected, as the funding goal ended up being the feature that led to the most informative splits in our decision tree. Whether or not a campaign was a staff pick was also a very important feature, as this provides extra visibility and credence to campaigns. Software and Web were found to be two very important categories, and name length also served as an important feature. Seen below are our evaluation metrics for the decision tree model.

```
Confusion Matrix:
[[959 330]
 [353 968]]
Accuracy: 0.7383141762452107
Precision: 0.7457627118644068
Recall: 0.732778198334595
F-1: 0.7392134402443682
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.74      0.74      1289
           1       0.75      0.73      0.74      1321

avg / total       0.74      0.74      0.74      2610

AUC: 0.820744328796214
```

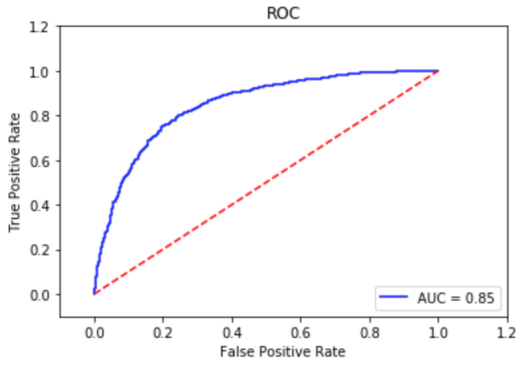


We obtained strong initial accuracy metrics, with accuracy, precision, recall, and F_1 all falling between .73-.75. In addition, our AUC of .82 shows that our initial model has strong discriminatory power.

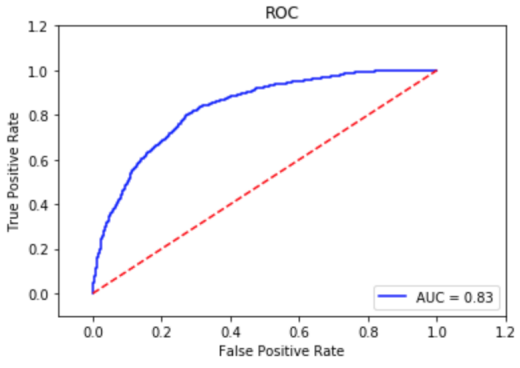
5.3.2 Random Forest

Next, we fit a random forest model again using the scikit-earn Python package. Like every other model, the estimator function RandomForestClassifier was fed through GridSearchCV to optimize parameters. As an ensemble method, the random forest model has the ability to provide better predictive power than a simple decision tree model. Instead of relying on one decision tree for prediction, a random forest model constructs multiple decision trees and uses the mode of the classes for classification. The optimal random forest classifier used 20 decision tree estimators, and had a minimum samples per leaf threshold of 10. Seen below are our evaluation metrics for the random forest model.

```
Confusion Matrix:
[[ 980  309]
 [ 280 1041]]
Accuracy: 0.7743295019157088
Precision: 0.7711111111111111
Recall: 0.7880393641180924
F-1: 0.7794833395731937
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.76      0.77      1289
           1       0.77      0.79      0.78      1321

avg / total       0.77      0.77      0.77      2610

AUC: 0.8469093576404081
```




```
Confusion Matrix:
[[ 939  350]
 [ 266 1055]]
Accuracy: 0.7639846743295019
Precision: 0.7508896797153025
Recall: 0.7986373959121877
F-1: 0.7740278796771826
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.73      0.75      1289
           1       0.75      0.80      0.77      1321

avg / total       0.76      0.76      0.76      2610

AUC: 0.8324311753385221
```



5.3.3 Logistic Regression

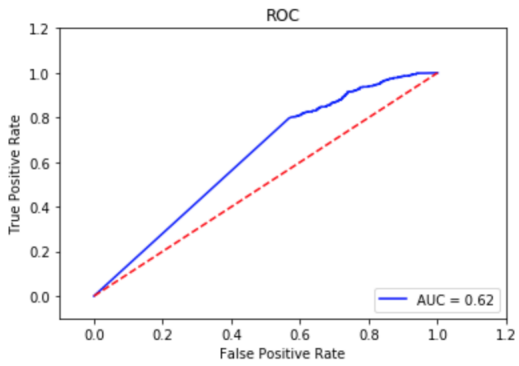
Logistic regression seeks to explain the relationship between one dependent binary variable and one or more independent variables of various types. Our logistic regression performed quite well when predicting positive with a precision of .75. However, the models ability to predict negatives was worse with an NPV of .73. Further, we achieved a recall of .798, a F_1 score of .774 and an AUC of .83.

Within our model, the following features were deemed most important for potential Kickstarter success: category_Shorts, staff_pick, country_HK, category_Festivals, and category_Plays. These seem to make intuitive sense given the nature of crowd sourced funding where projects with more tangible excitement value or are perceived as more feasible to accomplish (Shorts) thus getting more attention and donations. For the sake of spacing, our evaluation metrics for the logistic regression model can be seen under the random forest results.

5.3.4 Naive Bayes

We choose to fit a Naive Bayes model with a Gaussian distribution to our data. The Naive Bayes model is a learning algorithm based on applying Bayes Theorem along with the naive assumption that all data features are independent. The GaussianNB model within the Python package scikit-learn seemed like a good initial model to examine because of its core assumption that the features are normally distributed. Upon running our GridSearchCV, we found that the best naive Bayes model for the data included an alpha value of 0.5 with a prior fit; the alpha value is an additive Laplace smoothing parameter. Seen below are our evaluation metrics for the naive Bayes model.

```
Confusion Matrix:
[[ 427  862]
 [ 201 1120]]
Accuracy: 0.59272030651341
Precision: 0.5650857719475277
Recall: 0.8478425435276306
F-1: 0.6781713593702694
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.33      0.45      1289
           1       0.57      0.85      0.68      1321

avg / total       0.62      0.59      0.56      2610

AUC: 0.6229937824801837
```

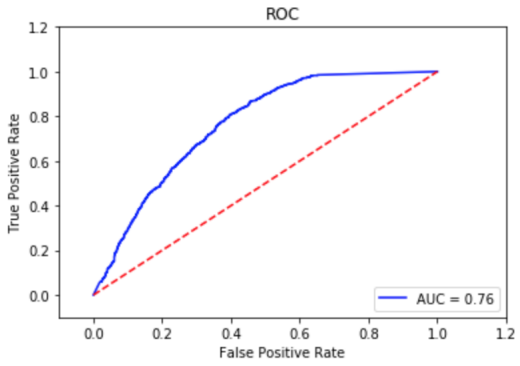


5.3.5 K-Nearest Neighbors

K Nearest Neighbors is an algorithm used for classification and regression where it utilizes the k closest training examples in order to help make predictions. This algorithm takes k points and uses them to vote on what outcome should be assigned to the centroid point. kNN also takes a number of specified inputs to allow flexibility in terms of number of neighbors as well as giving weights in reference to how much each neighbor counts in the voting process. The weights can either be specified as uniform, where each counts the same, or based on distance where closer points were given more value over further points. In our model, we specified the ideal number of neighbors to be 7 and to weight the points based on distance to the centroid. We chose k = 7 after running the kNN algorithm using k = (3, 5, 7, 9, 11) and selecting the k resulting in the best accuracy metrics.

Overall, our kNN model had slightly lower success than for some of the other models achieving accuracy metrics ranging from .64 to .70. With an AUC of .76, we considered this to be weak, but it did perform better than the Naive Bayes model. One of the major reasons we believe this algorithm had some trouble was the Curse of Dimensionality. Due to the number of features selected, our feature space became increasingly space which led to our algorithm to struggle obtaining the best classification results. This is a common problem, however, amongs algorithms that use non-linear decision boundaries leading to overfitting and poor generalization when applied to the validation set. Seen below are our evaluation metrics for the kNN model.

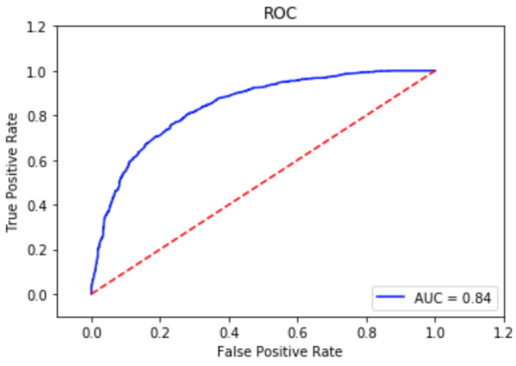
```
Confusion Matrix:
[[927 362]
 [475 846]]
Accuracy: 0.6793103448275862
Precision: 0.7003311258278145
Recall: 0.6404239212717638
F-1: 0.6690391459074733
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.72      0.69      1289
           1       0.70      0.64      0.67      1321

avg / total       0.68      0.68      0.68      2610

AUC: 0.7633075889918126
```



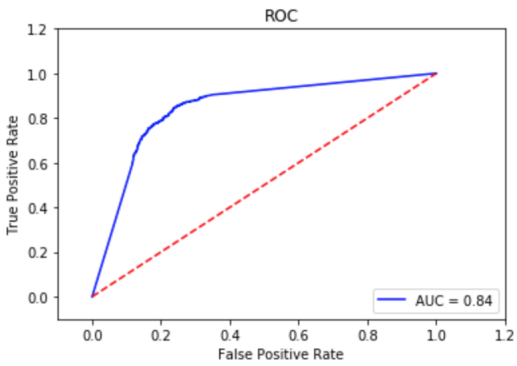
5.3.6 Adaptive Boosting

AdaBoost, or Adaptive Boosting, is an algorithm created by Yoav Freund and Robert Schapire in 2003 to help improve performance in conjunction with other algorithms. In our model, this methodology produced very consistent evaluation metrics ranging from .75 to .78, and did not display any particular weakness predicting positive or negative outcomes. Furthermore, our model used over forty estimators and found the size of the goal, launch_to_deadline_days, name_len_clean and blurb_len_clean to be the most important features. This fits well with what we have previously mentioned about Kickstarter campaigns. Having a feasible funding goal with plenty of time until the deadline are very important in generating wide interest from site visitors. Additionally, the length of the name and blurb attached to each project is also important to attaining funding dollars as the consumer needs to quickly understand and get excited about the task at hand. Seen below are our evaluation metrics for the AdaBoost model.

```
Confusion Matrix:
[[ 954  335]
 [ 293 1028]]
Accuracy: 0.7593869731800766
Precision: 0.7542186353631695
Recall: 0.7781983345950038
F-1: 0.7660208643815201
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.74      0.75      1289
           1       0.75      0.78      0.77      1321

avg / total       0.76      0.76      0.76      2610

AUC: 0.8422833631573043
```

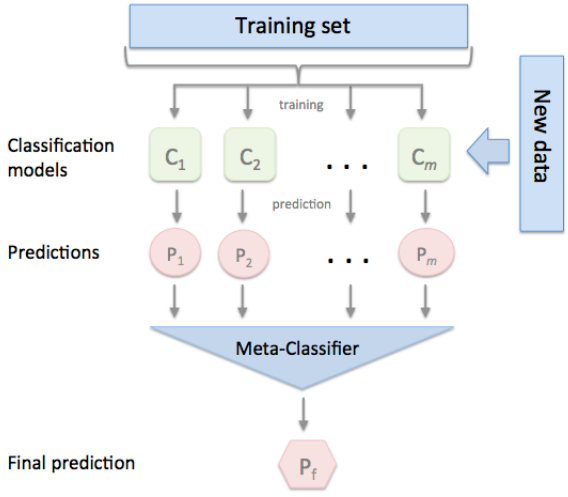



```
Confusion Matrix:
[[1002  287]
 [ 241 1080]]
Accuracy: 0.7977011494252874
Precision: 0.7900512070226774
Recall: 0.817562452687358
F-1: 0.8035714285714286
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.78      0.79      1289
           1       0.79      0.82      0.80      1321

avg / total       0.80      0.80      0.80      2610

AUC: 0.8433055217707158
```



5.3.7 TPOTClassifier

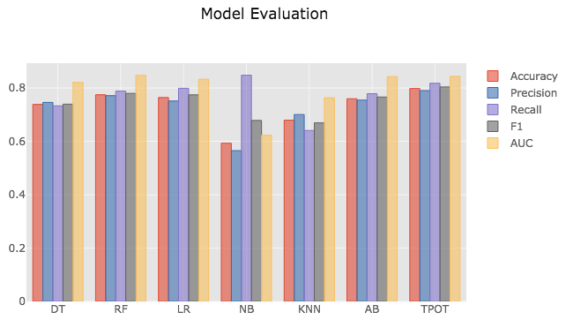
As described earlier in this paper, we used an automated machine learning tool called TPOT from EpistasisLab to find our seventh model. After 5 generations of intelligent model-hyper-parameter searches, the TPOT algorithm selected a stacking estimator, containing a base layer trained directly on features of the data under a stacked layer meta-estimator trained on the outputs from the base layer. The image below helps visualize this type of estimator.



For our project, we observed an interesting combination: a RandomForestClassifier stacked on top of predictions from a base RandomForestClassifier. Seen below are our evaluation metrics for the TPOT StackingEstimator model.

6. Conclusions

The below graph shows each model plotted against each other in terms of evaluation metrics. As we can see, the TPOT-generated classifier outperforms in most categories, except AUC; the random forest model seems to be the highest-performing discriminator in that category. Another quickly observable feature of the graph is spike in recall for the naive Bayes model.



Overall, we feel confident in our final results given the difficulty of prediction for this type of problem. Kickstarter campaigns are often challenging endeavors to get fully funded. Statista, a statistics portal based on consumer survey results, found that nearly 64% of all campaigns fail to meet their funding goals. Thus, we are pleasantly surprised with our models ability to be able to identify which bids will be labeled SuccessfulBool = 1 with a precision of .81.

7. Future Work

If we had more time to continue work, we identified a few different areas to explore that we felt would help continually improve our models predictive ability.

First, we wanted to dig deeper into the actual substance of the abstract and title for the Kickstarter campaign. This is valuable information that fully outlines the scope and direction of the project, but we were only able to analyze length of both features. We believe it would be very interesting to take this text data and use methodologies like creating a word cloud in order to identify key terms and phrases about the project. This allows us to get a better understanding of what the project actually entails beyond just the category and analyze whether there are higher success rates for certain key terms or project types.

Next, we felt it would be very interesting to be able to gather further data on other platforms like Twitter, Facebook and LinkedIn. Often, these projects will be started by individuals who may post additional information on these social media platforms as a way of further selling what they are trying to accomplish as well as generate further interest. For example, if an individual is already accomplished in a particular area and has a large social media following, we believe this would positively correlate with project success given the increased number of opportunities for contributions. Further, if they are posting additional information about the type of project, then this will be helpful as well in determining the potential interest level it may generate.

Finally, as true in most situations, we believe that if we continue to analyze more and more Kickstarter projects, then our models accuracy will further improve. With over 450,000 projects launched since the sites inception, our data set only scratches the surface and with further analysis, we are confident in improved prediction.

Acknowledgments

Downs, Rachel. Kickstarter Campaign Success. kickstarter_data_full, 21 Aug. 2017, data.world/rdowns26/kickstarter-campaigns.

Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore (2016). Automating biomedical data science through tree-based pipeline optimization. Applications of Evolutionary Computation, pages 123-137.

Spruyt, Vincent. The Curse of Dimensionality in Classification. Computer Vision for Dummies, 22 Mar. 2015.