Kickstarter Project Projection

Business Application of Machine Learning, UBC

I. BACKGROUND AND BUSINESS QUESTION

The dynamic domain of crowdfunding is witnessing an increasing need for predictive analytics to determine the success of projects before their official launch on platforms such as Kickstarter. Our client, a data analytics firm specializing in the crowdfunding space, is driven by the business imperative to anticipate the outcome of projects, enabling creators to make informed decisions and optimize their chances of success. This capability to predict is crucial, as it not only enhances the strategic deployment of resources but also fosters a more reliable and trustworthy crowdfunding environment.

The current exploring methods employed by the client—traditional methods of forecasting techniques that use simple historical patterns—are inadequate for the complex dynamics of Kickstarter's project success factors. In response to this gap, the client seeks to develop a sophisticated machine learning model that can analyze vast datasets, including project categories, funding goals, creator history, and market trends, to predict project success. This advanced model is envisaged to replace rudimentary approaches with a robust, data-driven framework, offering nuanced insights that can guide project creators toward achieving their funding objectives and backers toward more promising ventures.

Business Question:

- The crowdfunding landscape is growing, yet project creators face the challenge of predicting success before launch.
- Our client aims to build a machine-learning model to forecast the success of Kickstarter projects, providing strategic insights to creators and backers.
- Moving beyond basic trend analysis, the client's goal is to utilize comprehensive data to improve prediction accuracy, benefiting the entire Kickstarter community.

II. DATA AND STATISTICAL QUESTIONS

The dataset we used is from Kickstarter Projects in Kaggle to help Kickstarter predict if a project will be successful before it is released. The dataset consists of 378,661 observations starting from 2009-05-03 to 2018-03-03.

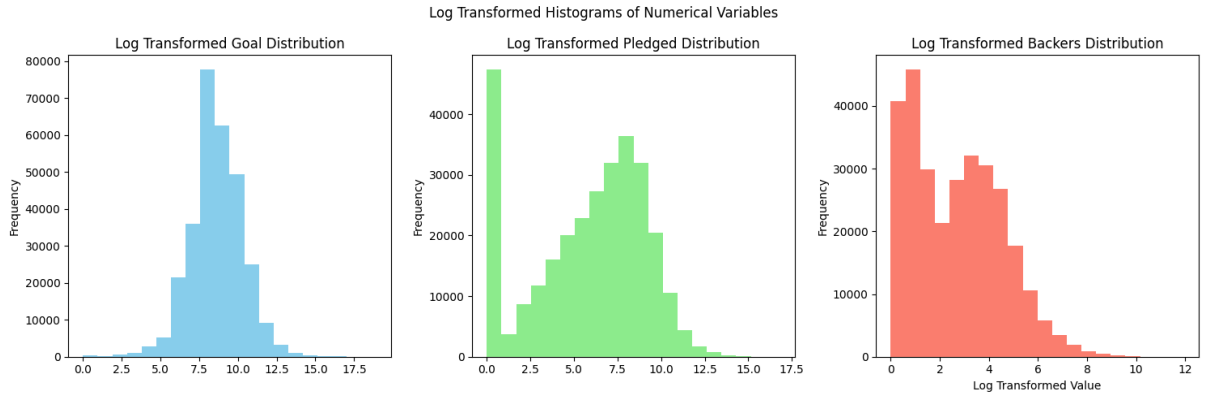
- The primary statistical question we aim to address is: "**How can we determine the likelihood of a Kickstarter project achieving its funding goal?**" This question is central to our business objective to develop a predictive model that can accurately

forecast whether a Kickstarter campaign will reach its funding target. This is a critical assessment tool for both project creators and potential backers.

- Limitations: Its reliance on historical data, which may not capture evolving trends or shifts in backer behaviour. Also, this scenario may not account for external variables like market conditions or global economic trends that also impact project success.
- Enhanced Model Predictability: We have introduced new variables that are anticipated to be indicative of a Kickstarter project's likelihood of success. These include temporal factors such as the time of launch, defined by the hour and day of the week, which may influence backer engagement and funding activity.
- Refined Statistical Inquiry: Our statistical analysis aims to parallel the core business question: What factors contribute to the success of Kickstarter campaigns? While our model focuses on predicting success probabilities, we note that it does not directly measure the potential impact or scale of success, which are elements that could align more closely with strategic business objectives.

III. EXPLORATORY DATA ANALYSIS

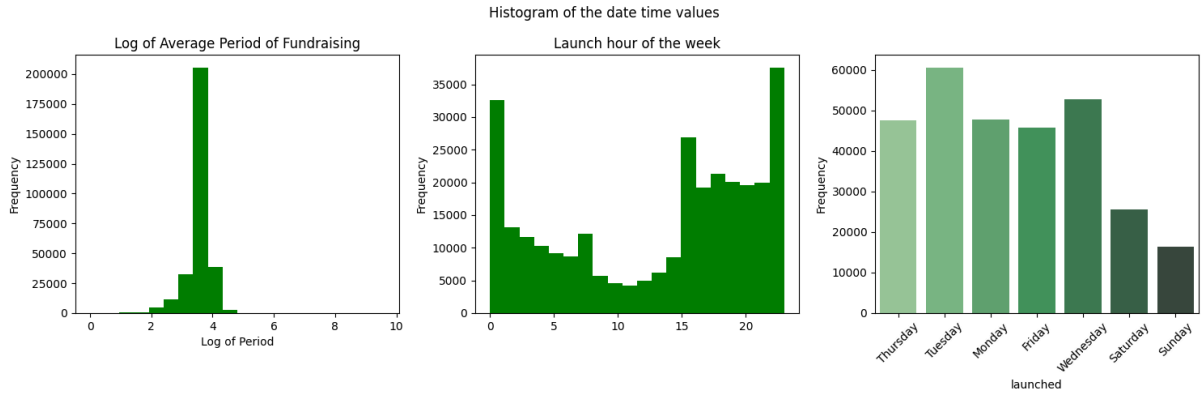
The charts given below represent the distributions of different columns of the dataset. This gives us an initial understanding of the data & how it could have an impact on our results.



Optimal Goal Setting: Analyzing the log-transformed goal distribution (blue histogram) helps businesses set realistic funding goals. Understanding the peak and spread of goal values allows project founders to choose attainable targets. Overambitious goals may deter potential backers, while too-conservative goals may limit project impact.

Pledged Amount Insights: The log-transformed pledged distribution (green histogram) reveals patterns in pledged amounts. Businesses can identify common pledge levels and tailor rewards or incentives accordingly. Insights from this distribution guide marketing strategies to attract backers.

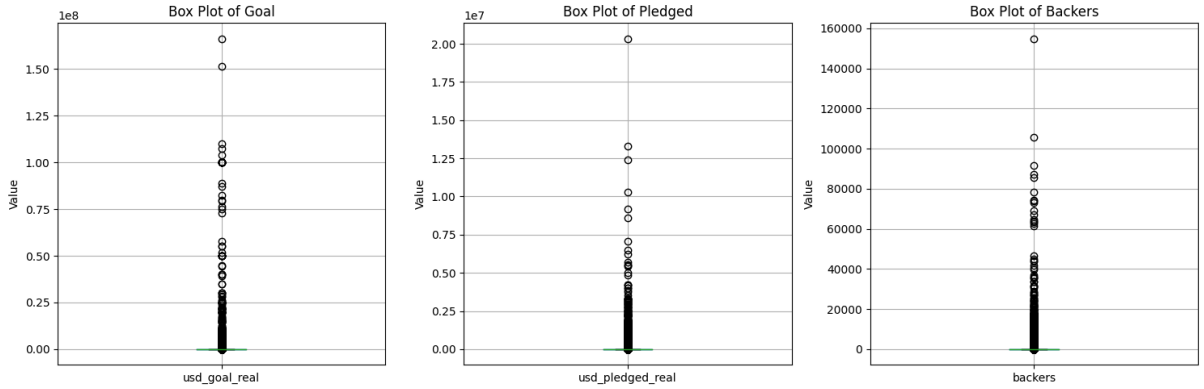
Backer Behavior Understanding: The log-transformed backer's distribution (red histogram) sheds light on backer engagement. Concentration at lower values suggests most projects have a limited number of backers. Businesses can optimize outreach efforts, engage existing backers, and attract new ones.



Log of Average Period of Fundraising: The histogram shows the distribution of fundraising periods (in logarithmic scale). A significant peak occurs around a log period value close to 2. This suggests that most fundraising campaigns last approximately 100 days (since 10^2 = 100).

Launch Hours During the Week: The second histogram represents the frequency of project launches at different hours of the week. Two prominent peaks are observed around 0 (midnight) and 20 (8:00 PM) hours. These hours are likely popular times for starting fundraising initiatives.

Days of the Week for Project Launches: The third histogram displays the days when projects were launched. Tuesday stands out as the most common day for launching fundraisers. Other days also contribute, but Tuesday has the highest frequency.

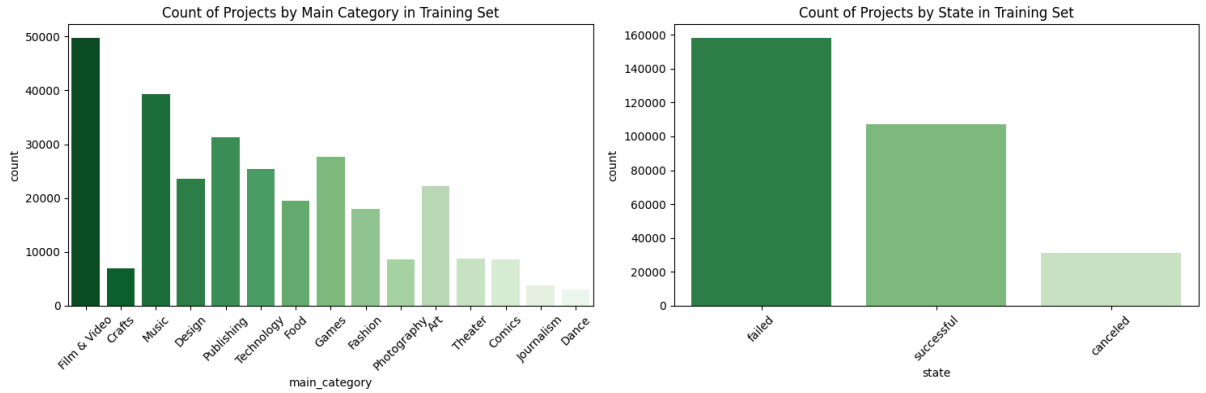


Each plot provides insights into different aspects of the project: goals, pledges, and backers. Box plots are useful for visualizing the distribution, central tendency, and variability of the dataset.

Box Plot of Goal: Represents the 'usd_goal_real' (goal amount in USD) on the x-axis and the 'value' (actual data) on the y-axis. The plot shows a concentration of data points around lower goal values, with some outliers at higher goals.

Box Plot of Pledged: Represents the 'usd_pledged_real' (actual pledged amount in USD) on the x-axis and the 'value' on the y-axis. Similar to the goal plot, but with fewer outliers at higher pledged amounts.

Box Plot of Backers: Represents the 'backers' (number of backers) on the x-axis and the 'value' on the y-axis. Indicates a significant concentration of backers at lower values, with several outliers extending upwards. These box plots provide a concise summary of the crowdfunding data, helping understand its distribution and identify potential outliers.



Count of Projects by Main Category:

Insight 1: Popular Categories The most common project categories are Film & Video, Crafts, and Music. Businesses can focus marketing efforts on these popular categories to attract more backers.

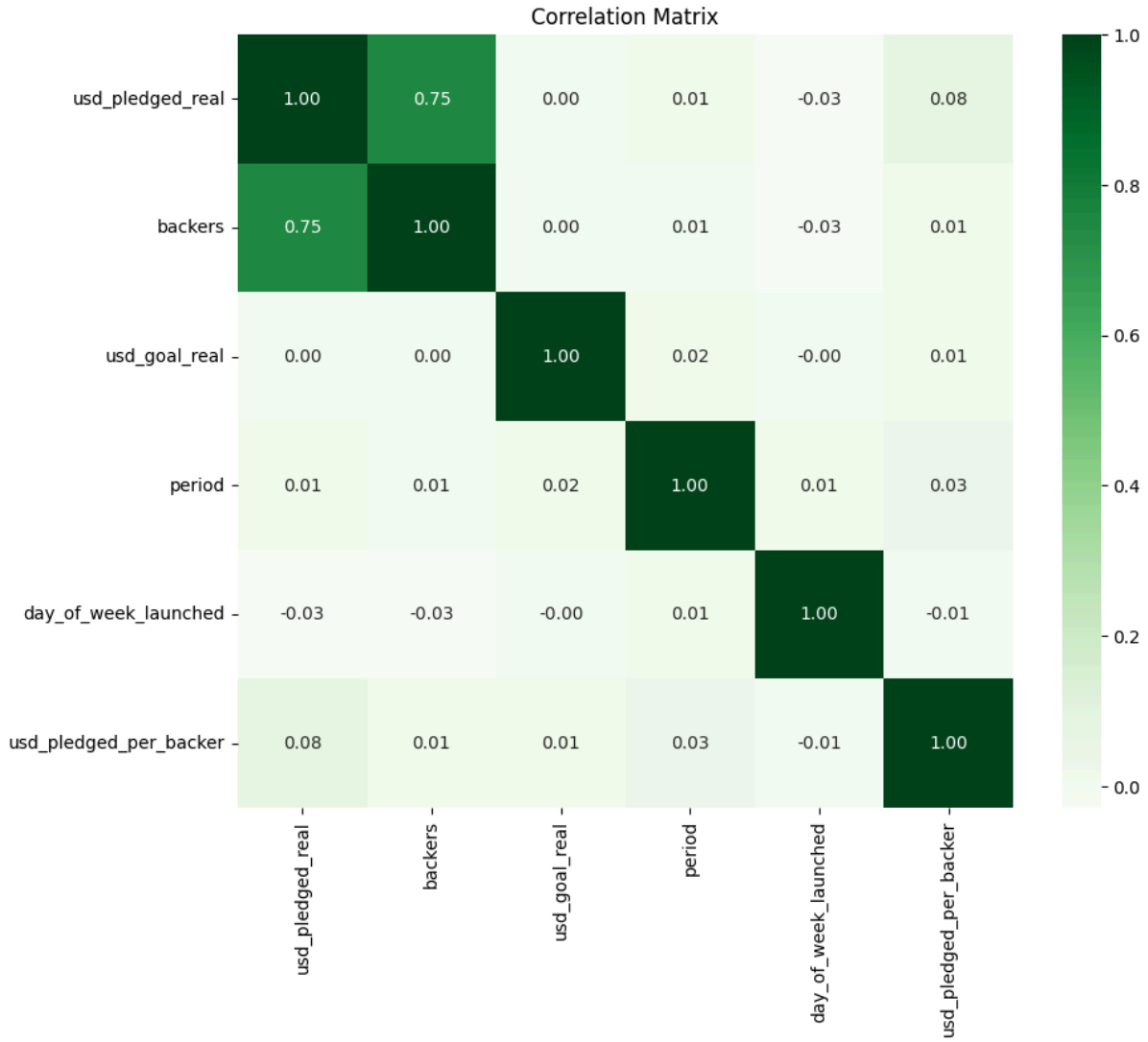
Insight 2: Niche Categories Categories like Dance, Journalism, and Comics have fewer projects. Businesses can explore niche markets within these categories for unique opportunities.

Count of Projects by State in Training Set:

Insight 1: High Failure Rate A significant number of projects have failed (over 140,000). Businesses should analyze why these projects failed and learn from their mistakes.

Insight 2: Success and Canceled Projects The number of successful projects is smaller but still substantial. Cancelled projects form a smaller portion. Businesses can study successful projects to identify winning strategies.

BIVARIATE ANALYSIS



Strong Positive Correlation: The correlation between usd_pledged_real (amount pledged) and backers is 0.75.

Insight: Projects with more backers tend to receive higher pledged amounts. Businesses can focus on strategies to attract and engage backers to increase funding.

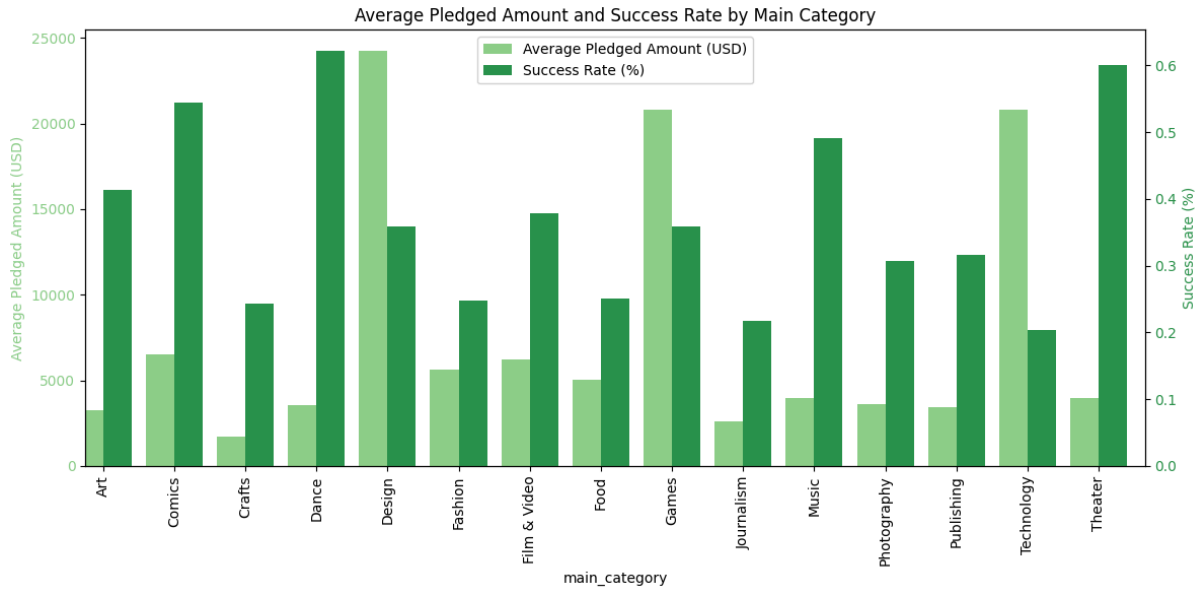
Weak Correlations with Goal and Time: The correlations between usd_goal_real (funding goal) and other variables are close to zero.

Insight: Funding goals do not strongly impact other variables in this dataset. Businesses should consider other factors beyond the goal amount.

Day of Week Launched: The correlation with other variables is not shown, but it's essential. **Insight:** Launching campaigns on specific days may influence success. Businesses can analyze which days yield better results.

Pledged per Backer: The correlation with other variables is not shown.

Insight: Understanding the average pledged amount per backer helps tailor rewards. Businesses can optimize reward tiers to encourage higher per-backer contribution.



Some major insights that can be drawn from the above chart are

Technology Category:

- **Average Pledged Amount:** Technology projects have a high average pledged amount.
- **Success Rate**: The success rate is moderate.
- **Insight:** While technology projects attract substantial funding, they don't always have the highest success rates. Businesses should focus on effective campaign management to convert pledges into successful outcomes.

Dance Category:

- **Average Pledged Amount**: Dance projects have a low average pledged amount.
- **Success Rate**: However, the success rate is one of the highest.
- **Insight:** Despite lower funding amounts, dance projects tend to succeed. Businesses can learn from their engagement strategies and replicate them in other categories.

Games Category:

- **Average Pledged Amount:** Games have both a significant average pledged amount and a high success rate.

- **Insight**: Games are a lucrative category with strong community support. Businesses can leverage this success by tailoring campaigns to gaming enthusiasts.

IV. MODELLING

- **Feature Pre-processing**

In the preliminary phase of developing the Kickstarter Success Model, we meticulously refined our dataset by excluding non-critical attributes. This included the removal of unique identifiers, project titles, and specific sub-categories, opting instead to focus on the more general 'main_category'. We also standardized our approach by converting all financial figures to a common currency, thereby eliminating the need for individual currency data. Furthermore, details such as project deadlines, launch dates, and initial commitment figures were omitted, considering their unavailability at the campaign's inception, to streamline our analysis and enhance model accuracy. We normalized the numerical data to ensure that no single attribute biased the results due to differences in scale. For categorical data, we apply one-hot encoding to convert the category labels into a format that our machine-learning algorithms can digest. This groundwork is critical in preparing the data for the heavy lifting of the modelling phase that follows.

- **Feature Selection**

We carefully designed the Kickstarter model by carefully selecting features that closely reflect how the platform works and the elements that may impact the success of a project. We focus on attributes that provide hard data on campaign performance and audience response, such as project categories, number of backers, geographies, funding amounts, goals, timing, and engagement metrics. These variables were chosen because of their proven impact on project outcomes, ensuring that our model is built on variables with meaningful links to success.

- **Reflection on Selected Performance Metrics**

Our choice of performance metrics for the Kickstarter prediction model is tailored to address the **classification challenges** posed by our dataset. We initially opted for accuracy but soon shifted to precision and recall to better evaluate true success predictions against the cost of false positives. We also adopted the F1 score for its balanced view of precision and recall, which is particularly useful for quick performance checks during model tuning. Including AUC-ROC helps us adjust model sensitivity to various operational thresholds, aligning with our business goals.

What's more, we're mindful of the potential impact on stakeholders, from project creators to backers, and the ethical implications of our model's predictions. To ensure fairness and avoid bias, we're committed to an ongoing evaluation of our metrics and model transparency.

- **Hyperparameter Tuning**

We choose models including Naive Bayes, Logistic Regression, Decision Tree, Random Forest, and KNN, using grid search and cross-validation to ensure robustness. For

Logistic Regression, we varied regularization strength to combat overfitting. Our Decision Trees and Random Forests were calibrated for optimal tree numbers and depth, as well as the minimum samples per leaf, to learn from the data while avoiding over-specialization. With KNN, the focus was on the neighbor count to accurately reflect data trends while ignoring noise. This meticulous tuning aims to hit the right complexity for our models to perform reliably on Kickstarter data, both in our current tests and future applications.

- **Model Selection**

Given the processed data, there are many choices of machine learning models to be chosen from. Intuitively, the biggest challenge is that the training set alone contains 378,661 samples and 15 features. Considering this, models requiring increased dimensional representation such as Supporting Vector Machines will be too expensive computationally ***(Appendix 1)***.

Therefore, we propose the following reasonable models for initiation:

1. **Baseline Model - Dummy Classifier**

We choose the Dummy Classifier as the baseline model to set the lowest possible performance. Based on our EDA analysis, we chose the 'stratified' strategy to generating predictions by respecting the class distribution of the training data. The results from the baseline model showed a test accuracy of 0.42723.

2. **Decision Tree:**

In our analysis of a Decision Tree classifier for the Kickstarter project, we constructed a preprocessing and classification pipeline that effectively managed both numerical and categorical data. Cross-validation with five folds indicated a solid accuracy of 82.22%, surpassing the baseline dummy classifier, though macro-averaged precision, recall, and F1 scores suggested room for improvement.

Our choice of a Decision Tree was informed by its known efficacy in handling large datasets. Hyperparameter tuning through GridSearchCV honed the model's settings, favoring the 'entropy' criterion and enhancing the model's score to approximately 89.16%. Despite the gains, the discrepancy between accuracy and other metrics hinted at class performance imbalances, pointing to areas for future refinement. The optimal model, featuring a max depth of 10 and a minimum of 2 samples per leaf, showcases the value of meticulous hyperparameter optimization in predictive accuracy.

Following the hyperparameter optimization process, our model achieved an impressive accuracy of 89.12% on the test dataset, signifying a notably high level of predictive performance. This concise evaluation illustrates the Decision Tree's strengths in our dataset and underscores the continuous need for model tuning and analysis, particularly in addressing class imbalance for more uniform classification success.

3. **Logistic Regression:**

The Logistic Regression Model is easy to implement because it does not require hyperparameter tuning. However, Logistic Regression makes strong assumptions of the data linearity and performs weakly for nonlinear data.

Within our analysis, we meticulously processed and transformed our data to optimize the performance of the Logistic Regression model. Numeric variables were standardized, utilizing a StandardScaler to negate the influence of differing scales, while categorical variables underwent one-hot encoding through a OneHotEncoder, ensuring they could be effectively utilized in our model. These transformations were systematically applied via a ColumnTransformer, which streamlined preprocessing within a comprehensive pipeline that also included our classifier, ensuring consistency and efficiency in our workflow.

The Logistic Regression algorithm exhibited a promising balance between accuracy and interpretability. After rigorous cross-validation, the model achieved an average accuracy of approximately 58%, indicating a robust predictive capability.

4. **Naive Bayes:**

Naive Bayes classifiers rely on the direct probability from the training data, which means they don't presume a specific data distribution beforehand, making them efficient and deterministic. Yet, they often fall short on model capacity and struggle with inference on new, unseen data.

For the Naive Bayes model, we treated feature preprocessing with the same meticulous care as we did for the Logistic Regression model. Numerical data were standardized, which is a critical step because Naive Bayes assumes that the numerical features are normally distributed. We also one-hot encoded categorical variables to convert them into a binary matrix, a necessary step for the Naive Bayes model to interpret these features correctly.

Despite its inherent simplicity, the Naive Bayes model achieved a training accuracy of approximately 42.95%, as seen from our Python output. It's crucial to note that while this figure might not seem overly impressive, Naive Bayes is known for better performance on unseen data compared to the training set, especially when the assumption of independence holds. The model showed a relatively high recall for successful projects, which could be particularly beneficial for identifying projects with a strong chance of success. However, precision for the 'canceled' class was low, indicating a tendency of the model to incorrectly label projects as canceled.

5. **Random Forest**:

Because our decision tree model generated a high score of 89.12%, we started to think about developing a Random Forest model. Random Forest generally provides high accuracy because it combines the predictions of multiple decision trees to produce a more robust and stable prediction. It is less prone to overfitting than individual decision trees, as it averages the results of many trees, which individually might overfit the data.

We introduced a Random Forest Classifier, a powerful ensemble learning technique, to further enhance our predictive capabilities in forecasting the success of Kickstarter projects. The Random Forest model excels in capturing complex relationships within the data by aggregating predictions from multiple decision trees. We encapsulated this model within a pipeline that seamlessly integrates data preprocessing steps and the Random Forest Classifier.

The hyperparameter tuning was performed through an exhaustive grid search, exploring various combinations of hyperparameter values. The grid search involved tuning parameters such as the number of estimators (trees) in the forest, the maximum depth of each tree, and the minimum number of samples required to split or form a leaf node. Through cross-validation with 3 folds, we systematically evaluated the model's performance on different subsets of the training data.

Upon completion of the grid search, we accessed the best-performing hyperparameters and the corresponding cross-validated accuracy score. The optimal configuration includes 100 or 200 estimators, with consideration for maximum depth, minimum samples split, and minimum samples leaf. This model achieved a cross-validated accuracy score that represents its estimated accuracy on unseen data, providing valuable insights into its predictive capabilities.

With its ability to handle complex relationships and capture feature importance, the Random Forest Classifier is a robust candidate for improving our project success predictions. As we move forward, we will leverage these findings to refine our model and contribute to the overarching goal of fostering a more reliable and insightful crowdfunding environment on Kickstarter.

6. KNN model

We discussed the K-Nearest Neighbors (KNN) model, a versatile algorithm that can be applied to classification and regression problems. It's particularly well-suited for nonlinear classification due to its simplicity and the lack of an explicit training phase. The core idea is to predict the label of a new point by looking at the 'k' closest labeled points from the training data, taking a majority vote for classification or an average for regression.

Thus, we decided to develop a KNN model to utilize its simple and instance-based learning algorithm for our classification tasks. Since KNN's approach is straightforward to implement. The theoretical foundation is also well-established, making it a go-to method for many practitioners. So we focused on our selection of k-values. We constructed a grid of hyperparameters and performed an exhaustive search to find the best possible combination of parameters of the KNN model.

We initially set the range of k-value from 1 to 5, which gave us the best value with the k-value of 5. Then we increased to a larger range of grid search of the potential k values from 1 to 8 and decreased the cross-validation to 3 considering the duration of running the model. With the new parameter, it took us about 3 hours to run the model with an i7-12700 CPU and provided us with the best k-value of 7.

The final accuracy score of our KNN model was around 67% with a k-value of 7 ***(Appendix 2)***, meaning that it had a prediction that was better than the dummy model but still lacked accuracy. The result made sense because KNN is a non-parametric and lazy learning algorithm, meaning that it does not make any assumptions on the underlying data distribution and it does not use the training data points to do any generalization.

V. Communication of Results, and Advice to a Non-expert

Model	Test Score
DummyClassfier	42.72%
Decision Tree	89.12%
Logistic Regression	58.06%
Naive Bayes	42.95%
Random Forest	89.12%
KNN	67%

We tested several machine learning models to find the one that performs the best in predicting project success. We used different algorithms, like Dummy Classifier, Logistic Regression, Naive Bayes, Random Forest, KNN and a Decision Tree. These are like different tools in a toolbox, each with its strengths and weaknesses. Our standout performer is the Decision Tree model & Random Forest. It's like having a seasoned expert analyzing all aspects of a project and providing a forecast of its success.

Our chosen model for this predictive analytics task is a Decision Tree Classifier & Random Forest, a sophisticated algorithm that analyzes various project attributes to make predictions about their success. Unlike traditional forecasting methods, our machine learning models take into account a multitude of factors, including project category, funding goals, creator history, and market trends, to provide nuanced and accurate predictions.

The application of this machine learning model holds significant potential for project creators and backers within the Kickstarter community. Creators can leverage the model's predictions to optimize their project launch strategies, increasing the likelihood of success. For backers, the model serves as a reliable tool to identify and support ventures with a higher probability of achieving their funding objectives.

Recommendations to Non-Experts:

Utilize Prediction Insights: Creators can incorporate the predictions provided by our model into their project planning. Understanding the factors influencing success can guide decisions related to project category, funding goals, and overall marketing strategy.

Enhance Decision-Making: Backers can make more informed decisions by considering the predictions generated by our model. This can lead to increased confidence in supporting projects that align with their preferences and have a higher chance of success.

VI. REFERENCES

1) https://towardsdatascience.com/
2) https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac
3) https://medium.com/@data.science.enthusiast/logistic-regression-tune-hyperparameters-python-code-fintech-does-it-bring-any-value-619e172565e6#:~:text=Hyperparameters%20are%20the%20parameters%20that,for%20a%20logistic%20regression%20model.
4) https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/

Appendix

Our appendix is mainly used for showing the results of the model running for an extremely long time and working not well in our case.

Appendix 1

SVC Model - taking more than 24 hours to run and still not able to generate the results

SVC Model

In [32]:
```
svc_pipe = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("svc", SVC())
])
```

In [*]:
```
param_grid = {
    'svc__gamma': [0.1, 1, 10, 100],
    'svc__C': [0.1, 1, 10, 100]
}
random_search = RandomizedSearchCV(
    svc_pipe,
    param_distributions=param_grid,
    cv = 2,
    verbose = 2,
    random_state = 289,
    n_iter = 2
)


random_search.fit(X_train,y_train);
```

```
Fitting 2 folds for each of 2 candidates, totalling 4 fits
[CV] END ..........................svc__C=1, svc__gamma=10; total time=79.7min
[CV] END ..........................svc__C=1, svc__gamma=10; total time=90.8min
[CV] END ..........................svc__C=10, svc__gamma=1; total time=119.7min
[CV] END ..........................svc__C=10, svc__gamma=1; total time=116.2min
```

In [*]:
```
random_search.score(X_test, y_test)
```

Appendix 2
KNN Model - taking more than 3 hours to run and showed a low accuracy

knn

In [38]:
```
from sklearn.neighbors import KNeighborsClassifier
```

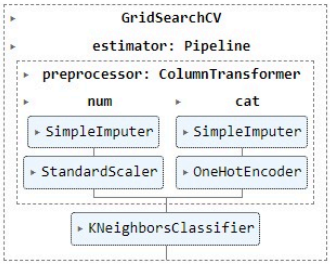
In [39]:
```
knn_pipe = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("knn", KNeighborsClassifier())
])
```

In [40]:
```
param_grid = {
    'knn__n_neighbors': list(range(1, 9))
}

grid_search = GridSearchCV(knn_pipe, param_grid, cv=3,n_jobs=-1)
```

In [41]:
```
grid_search.fit(X_train, y_train)
```

Out[41]:



In [42]:
```
best_score = grid_search.best_score_
best_param = grid_search.best_params_
print(f"The best score from GridSearchCV is: {best_score} with k value of {best_param}")
```

```
The best score from GridSearchCV is: 0.6729213802642695 with k value of {'knn__n_neighbors': 7}
```