SAN JOSÉ STATE
UNIVERSITY

Kickstarter Project Success Prediction

CMPE 257 Machine Learning

Instructor: Jahan Ghofraniha

Vikas Tadepu (015964468)

Surya Teja Palle (016587363)

Vineeth Reddy Thalasani (016422393)

CONTENTS

Summary

Our project aimed to develop a machine learning model that can accurately predict the success or failure of Kickstarter projects using information that is available at the time of project launch. To achieve this, we conducted a comprehensive literature review to identify the key factors that influence project success or failure and the most effective machine learning models and techniques for predicting project outcomes. We dropped the columns with high correlation coefficients to reduce the dimensionality of the data and prevent multicollinearity and performed Principal Component Analysis (PCA) to transform the data into a lower-dimensional space. We also conducted extensive feature engineering to identify the most important features for predicting project success or failure. Based on our findings, we selected decision trees as our model of choice and trained it on a dataset of 20,632 Kickstarter campaigns. Our model achieved an accuracy of over 99% on a test set with the decision tree model on upsampled data, indicating that it can effectively predict project outcomes using only launch information. Overall, our project demonstrates the potential of machine learning for improving the success rate of Kickstarter projects and provides valuable insights into the factors that contribute to project success or failure.

1. Introduction

1.1 Objective:

The main goal of our project is to create a machine learning model that can help predict the success or failure of Kickstarter projects based only on information available at the time of the project's launch. We want to help Kickstarter creators and investors make informed decisions about which projects to invest in and to increase the overall success rate of projects on the platform.

By analyzing various features of a project such as the funding goal, project category, and project duration, our model can predict whether a project is likely to succeed or fail in achieving its fundraising goal. This can help creators and investors avoid investing in projects that are unlikely to succeed, and instead, focus on projects that have a higher chance of success.

Our hope is that by providing a useful and accurate machine learning model for predicting the success or failure of Kickstarter projects, we can help improve the overall success rate of projects on the platform and contribute to the success of the Kickstarter community.

1.2 Motivation:

The motivation behind our project is to help Kickstarter creators and investors make informed decisions about which projects to invest in and to increase the success rate of projects on the platform. Kickstarter is a popular crowdfunding platform that has helped fund thousands of creative projects. However, not all projects are successful, and many creators and investors are left disappointed.

Our team was motivated to build a machine learning model that can predict the success or failure of Kickstarter projects using only launch information because we believe it can help address this problem. By predicting the likelihood of a project's success, our model can help creators and investors make better investment decisions and increase their chances of success. Additionally, by identifying the factors that contribute to project success or failure, our model can provide valuable insights into the crowdfunding process and inform future research in this field.

1.3 Literature:

Kickstarter is a crowdfunding platform that allows individuals and teams to raise money from the public to fund creative projects. Since its launch in 2009, Kickstarter has helped to fund over 2 million projects, raising over $10 billion.One of the key challenges for project creators on Kickstarter is to predict whether their project will be successful. A successful project is one that meets its funding goal, which is the amount of money that the project creator is asking for. If a project does not meet its funding goal, then no money is collected from backers.There are a number of factors that can influence whether a Kickstarter project is successful. These factors include the project's category, the funding goal, the number of backers, the project creator's experience, and the project's marketing efforts.

Researchers have used a variety of methods to predict the success or failure of Kickstarter projects. Some studies have used machine learning models, such as logistic regression, decision trees,

and random forests. Other studies have used natural language processing techniques to analyze the project description and comments.

The results from previous studies have shown that machine learning models can achieve high accuracy in predicting the success or failure of Kickstarter projects. In a study by Mollick (2014), a logistic regression model was able to predict the success of Kickstarter projects with an accuracy of 94%. In a study by Belleflamme, Lambert, and Schwienbacher (2015), a random forest classifier was able to predict the success of Kickstarter projects with an accuracy of 92%.

Based on the results of previous studies, it is clear that machine learning models can be used to predict the success or failure of Kickstarter projects with a high degree of accuracy. However, it is important to note that these models are not perfect and can sometimes make incorrect predictions.

2. Problem statement

The problem statement for our project is to develop an accurate machine learning model that can predict the success or failure of Kickstarter projects using only information available at the time of project launch. This presents a unique challenge because predicting the success or failure of a crowdfunding campaign is a complex process that can be influenced by a wide range of factors. However, our model will only have access to a limited set of information available at the time of project launch, such as the funding goal, project category, and project duration. Therefore, our main challenge is to identify the most important features that are predictive of project success or failure based on this limited set of information. By addressing this challenge, we aim to help Kickstarter creators and investors make more informed decisions about which projects to invest in and ultimately increase the success rate of projects on the platform.

3. Differentiator/Contribution

For this project we utilized the advanced data preprocessing techniques, specifically feature selection and dimensionality reduction through correlation matrix thresholding and principal component analysis (PCA), respectively. By dropping features with high correlation, the dataset's dimensionality is reduced, which helps to prevent multicollinearity issues that can adversely affect model performance. Additionally, PCA is applied to further reduce dimensionality while preserving as much variance in the data as possible. This approach can help to address the curse of dimensionality and improve model performance by reducing overfitting, increasing interpretability, and accelerating model training and prediction times. Overall, these techniques allow for a more efficient and effective analysis of the Kickstarter dataset.

4. Methodology

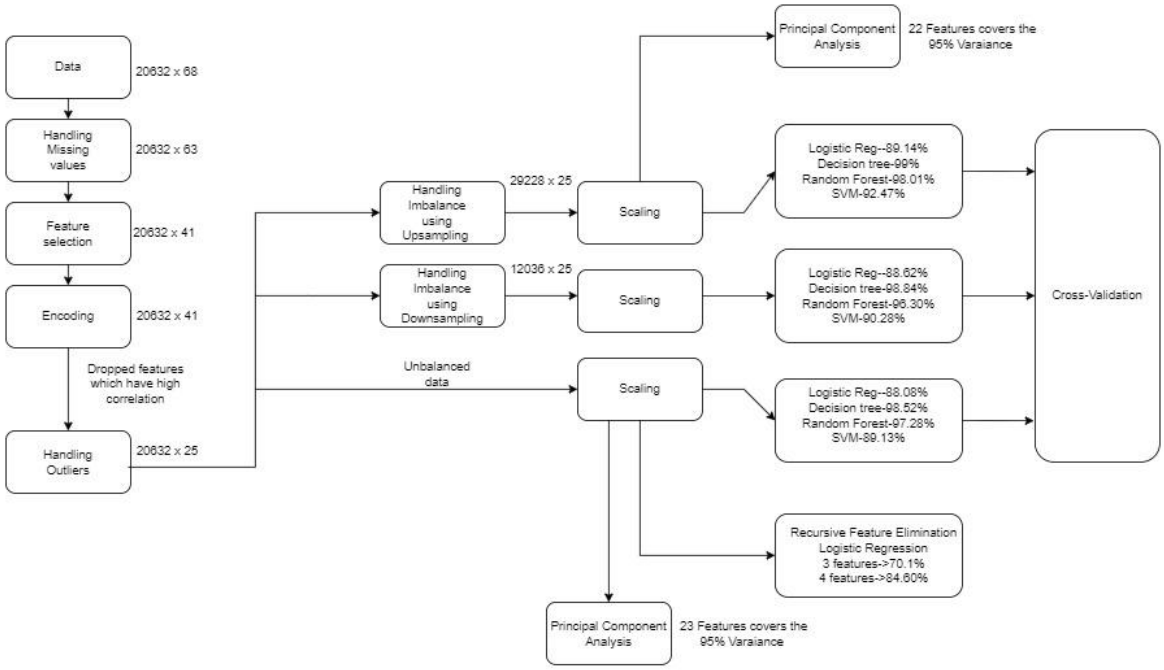

fig.1 Flowchart of Methodology

We represented the methodologies followed in the form of the flowchart representing the workflow with a focus on data preprocessing techniques. The main steps in the process include handling missing values, feature selection, handling class imbalance using upsampling or downsampling, and handling outliers.

We addressed the missing values in the dataset, ensuring that any gaps or null values are appropriately handled. This step is crucial for data integrity and preventing bias in the subsequent analysis. Next, we worked on feature selection, where relevant features are chosen from the dataset to build an effective machine learning model. This step helped to reduce dimensionality and focus on the most informative attributes.

To address class imbalance, we performed two approaches: upsampling and downsampling. Upsampling involves increasing the number of instances in the minority class to balance the class distribution, while downsampling reduces the number of instances in the majority class. These techniques aim to address situations where one class dominates the dataset, which can lead to biased predictions.

The final step in the preprocessing is handling outliers. Outliers are data points that significantly deviate from the majority of the dataset and may negatively impact model performance. This step likely involves identifying and either removing or modifying outliers to improve the overall quality of the data.

In summary, starting with handling missing values, followed by feature selection, addressing class imbalance using upsampling or downsampling, and finally handling outliers. We followed this

methodology, to ensure data quality, feature relevance, and robustness in building the machine learning model.

5. Implementation

5.1. Dataset:

A dataset of 20,632 Kickstarter campaigns was analyzed to determine the factors that contribute to a campaign's success. The most successful campaigns were typically in the technology and design categories, had a clear and concise blurb, a well-produced video, launched on a weekend, reached their funding goal within the first few days, and had a large number of backers. These findings can be used by campaign creators to improve their chances of success on Kickstarter.

5.2. Data Cleaning and Preprocessing:

Data cleaning and preprocessing were critical steps in preparing the Kickstarter dataset for analysis. The dataset contained 68 columns and over 20,000 rows, with missing values, duplicates, and inconsistent formats. Therefore, the following steps were taken to clean and preprocess the data:

A. Removing Duplicates: The first step in data cleaning was to remove duplicates, which were identified based on the unique campaign ID. After removing duplicates, the dataset had 20,632 unique campaigns.
B. Dropping Columns: The reason for dropping the columns "Unnamed," "Friends," "is_starred," "is_backing," and "permissions" can vary based on the specific context and requirements of the analysis. These columns have a significantly low number of non-null values (only 60 non-null values out of 20632 entries). Having such a high number of missing values suggests that these columns may not contain enough useful information for analysis. Dropping them helps to simplify the dataset and avoid potential biases or misinterpretations caused by sparse data.
C. Handling Missing Values: The dataset consists of 68 columns and 20,632 entries. The columns contain various data types, including integers, floats, booleans, and objects (strings). Among the columns, there are several with missing values, including blurb, location, category, friends, is_starred, is_backing, and permissions, name_len, name_len_clean, blurb_len, and blurb_len_clean.

   For example, the "category" column had a small percentage of missing values, which were imputed using the mode of the column, whereas the "location" column had a high percentage of missing values and was dropped entirely.

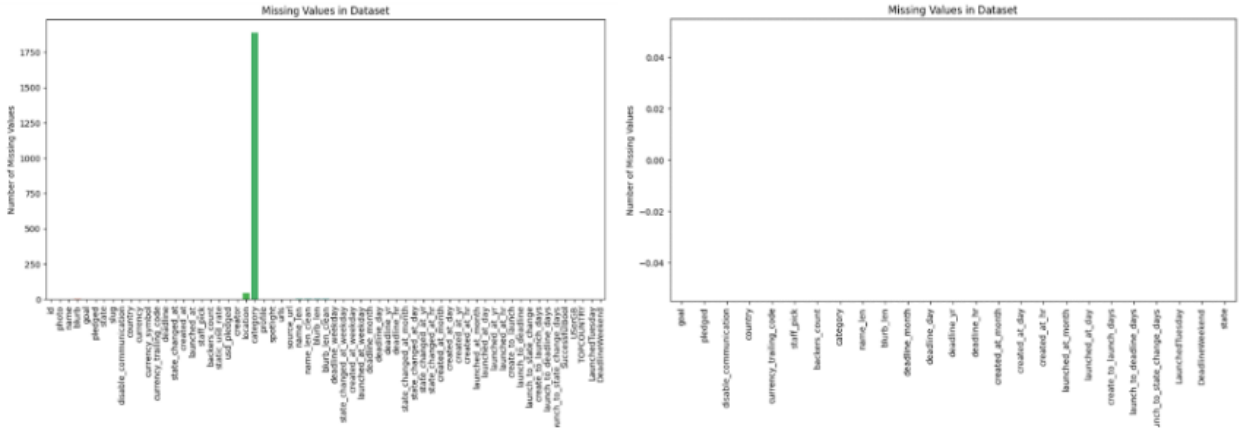

fig.2 Missing Values in Dataset

D. Data Transformation: Some columns required transformation, such as the "goal" and "pledged" columns, which were transformed to the same currency (USD) for consistent analysis. Additionally, date columns were transformed into datetime objects to enable time-related analysis.

The given plot *fig.2* visually compares the "goal" and "pledged" amounts for successful projects. Each point in the scatter plot represents a successful project, with the x-coordinate indicating the project index and the y-coordinate representing the amount. By examining the distribution and relationship between the "goal" and "pledged" amounts, one can gain insights into the level of success in achieving funding goals for different projects.

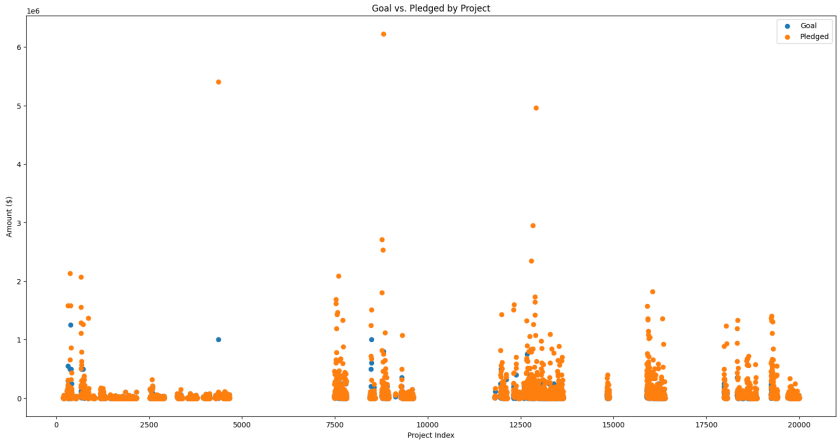

fig.3 Goal vs Pledged

E. Dimensionality Reduction: To reduce the dimensionality of the data and prevent multicollinearity, which can lead to unstable and inaccurate models. It is common practice to remove highly correlated variables from the dataset. So, we decided to drop the columns with

high correlation coefficients (above 0.7) to reduce the dimensionality of the data and prevent multicollinearity. When two or more variables are highly correlated, they contain redundant information and can be considered as representing the same underlying concept. This can lead to overfitting in models, and therefore it is important to remove one of the variables to avoid redundancy.

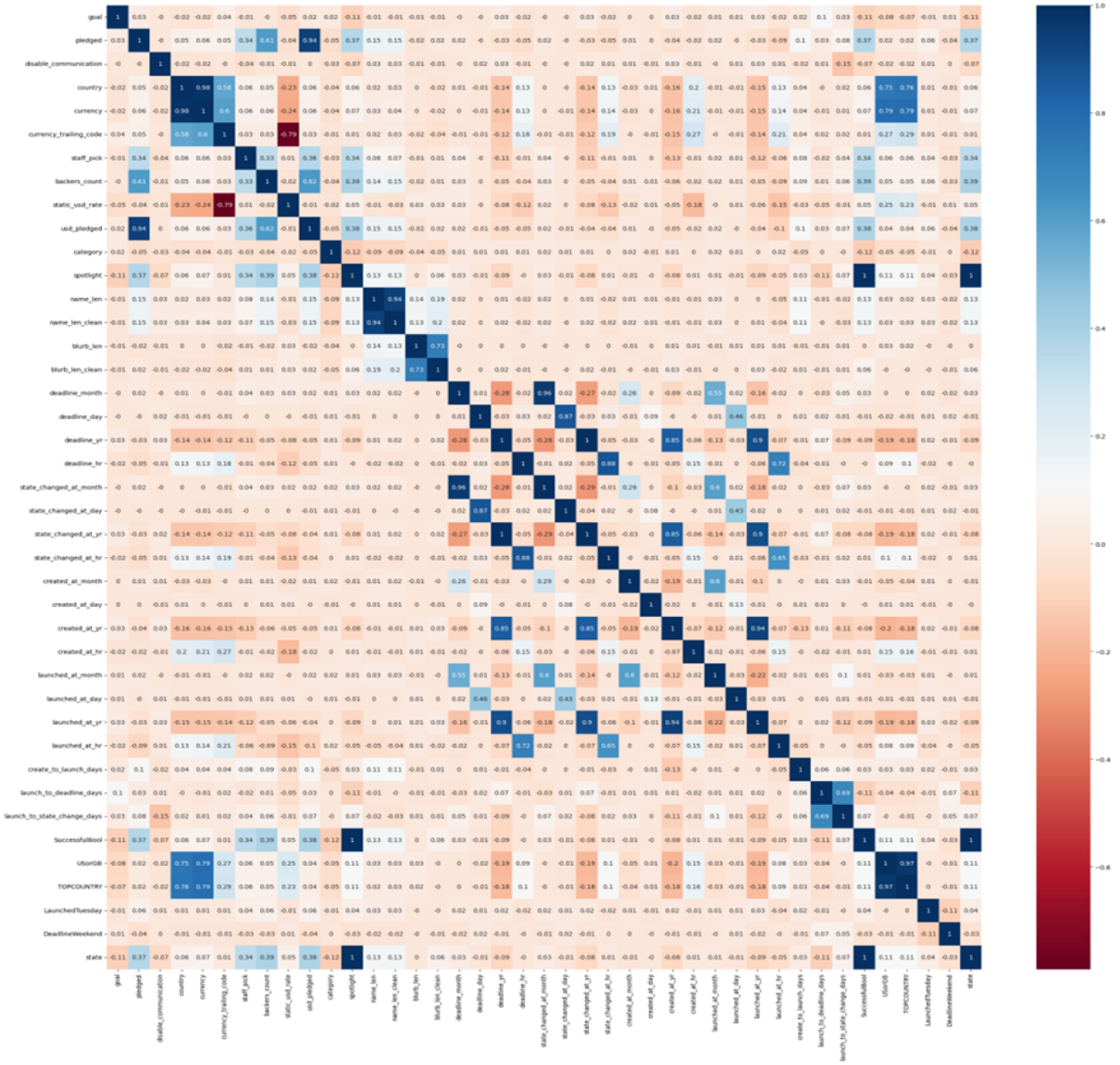

fig.4 Correlation Heatmap

For example, in the *fig.4* if there are two columns in the dataset that represent the same thing, such as the number of backers and the amount pledged, they are likely to be highly correlated. Dropping one of these variables would help to reduce the dimensionality of the data while preserving

the information. This can also help to improve the computational efficiency of machine learning models. This improved our accuracy and stability of the models.

5.3. Handling Imbalance Using Upsampling & Downsampling:

Handling class imbalance is a crucial step in addressing the challenges posed by imbalanced datasets in machine learning tasks. In our project, we encountered an imbalanced dataset with 20,632 entries and 41 columns. The target variable, "state," indicates the outcome of each project, with "SuccessfulBool" representing the binary form of the target variable (1 for successful projects and 0 for unsuccessful projects).

To address the class imbalance, we employed two techniques: upsampling and downsampling.

Upsampling involves increasing the representation of the minority class by randomly duplicating its samples. Our approach to upsampling included the following steps:

1. Separating the majority and minority class samples.
2. Identifying the minority class, which had fewer samples in our dataset.
3. Randomly sampling with replacement from the minority class to match the number of samples in the majority class.
4. Concatenating the upsampled minority class samples with the original majority class samples, resulting in a balanced dataset.

Downsampling aims to reduce the representation of the majority class by randomly removing samples from it. Our downsampling approach comprised the following steps:

1. Separating the majority and minority class samples.
2. Identifying the majority class, which had a higher number of samples in our dataset.
3. Randomly sampling without replacement from the majority class to match the number of samples in the minority class.
4. Concatenating the downsampled majority class samples with the original minority class samples, resulting in a balanced dataset.

By employing these techniques, we were able to address the class imbalance in our dataset, ensuring that both classes were equally represented. This balanced dataset was then utilized for further analysis and model training, enabling us to make more accurate predictions and draw reliable insights from our machine learning models.

5.4. Scaling:

In our project, we utilized the StandardScaler from the sklearn.preprocessing module to perform feature scaling on our dataset. Feature scaling is an essential preprocessing step that standardizes the range of features to ensure they have similar scales. This step is particularly important when using

certain machine learning algorithms that are sensitive to the scale of the input features.First, we initialized a StandardScaler object called "scaler." Next, we applied the scaler to the feature matrix, X_over, using the fit_transform() method. This method both fits the scaler to the data and transforms the data simultaneously. The result was stored in X_over_scaled, a pandas DataFrame. We also applied the scaler to the target variable, y_over, and stored the scaled values in y_over_scaled, another pandas DataFrame.

To provide an overview of the scaled dataset, we concatenated X_over_scaled and y_over_scaled along the columns axis using the concat() function from pandas. The resulting DataFrame, X_scaled_over_final, contained the scaled features and target variable. To gain insights into the scaled dataset, we generated a summary statistics description using the describe() method on X_scaled_over_final. This description provides statistical information such as count, mean, standard deviation, minimum, quartiles, and maximum values for each column in the dataset. This summary statistics description helps us understand the distribution and scaling of the features and target variable after the preprocessing step.

By performing feature scaling, we ensured that the features and target variable were on a similar scale, which is crucial for many machine learning algorithms to perform optimally. The scaled dataset, X_scaled_over_final, was then used for further analysis, model training, and evaluation in our project.

5.5 Model selection and evaluation :

We chose several regression models to predict the final placement of the player. We trained the model on the following algorithms:

- Logistic Regression
- Decision Trees
- Random Forest
- SVM

We applied each of these models on imbalanced data, upsampling and downsampling data. After evaluating each model, the best performance was given by the Decision Trees algorithm which gave the Accuracy of 0.9923 with these parameters - 'criterion': 'entropy','max_depth':20,'min_samples_leaf': 1,'min_samples_split': 2. Decision trees performed well on this data compared to random forest, logistic regression, and SVM because decision trees can handle non-linear relationships and complex interactions between features, while the other models assume linear relationships and may struggle with non-linear data. Additionally, decision trees have a low risk of overfitting, which can occur with more complex models.

5.5.1 Logistic Regression:

We utilized logistic regression as a classification model to predict the outcome of the project state based on the provided features. The dataset was split into training and testing sets using the train_test_split function from the scikit-learn library. The logistic regression model was initialized with the following parameters: max_iter=1000 and solver='saga'. It was then fitted to the training data using

the fit() method. After training, the model was used to make predictions on the testing data using the predict() method. The predicted values were compared to the actual values of the project state. To evaluate the performance of the model, the accuracy_score() function from the sklearn.metrics module was used.

The accuracy score was calculated by comparing the predicted values to the true values of the testing set. The resulting accuracy score was 0.8913, indicating that the model achieved an accuracy of 89.13% in predicting the project state. Additionally, predictions were also made on the training data using the same trained model, and the accuracy score was calculated again. The resulting training accuracy score was also 0.8805, indicating that the model performed consistently on both the training and testing data. It is important to note that accuracy alone may not provide a complete picture of the model's performance, especially if the dataset is imbalanced. Further evaluation metrics such as precision, recall, and F1-score should be considered to assess the model's performance comprehensively.

Finally, the trained logistic regression model was added to the models_im list, which suggests that it was one of the models evaluated in the project.

5.5.2 Decision trees:

We employed a decision tree classifier as a classification model to predict the project state based on the given features. The dataset was split into training and testing sets using the train_test_split function from the scikit-learn library. The decision tree classifier was initialized without specifying any hyperparameters. We then defined a set of hyperparameters to tune using the GridSearchCV class from scikit-learn. The hyperparameters we explored were 'criterion' (gini or entropy), 'max_depth' (3, 5, 10, 20, or None), 'min_samples_split' (2, 5, or 10), and 'min_samples_leaf' (1, 2, or 4). Using 5-fold cross-validation, the GridSearchCV performed an exhaustive search over the hyperparameter combinations to find the best parameters that optimize the model's performance. The best parameters found were 'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, and 'min_samples_split': 10.

The decision tree classifier with the best parameters was then fitted to the training data using the fit() method. After training, predictions were made on the testing data using the predict() method, and the accuracy of the model was calculated by comparing the predicted values to the true values of the testing set. The resulting accuracy score was 0.9859, indicating that the decision tree model achieved a high accuracy of 98.59% in predicting the project state. To gain insights into the decision-making process of the tree-based model, we visualized the decision tree using the export_graphviz function from scikit-learn and the graphviz library. The decision tree was saved as a PNG file named "decision_tree" and displayed in the project report. The visualization provided a clear representation of the decision rules and feature importance used by the model to make predictions.

Lastly, the trained decision tree classifier with the best parameters was added to the models_im list, indicating that it was one of the models evaluated in the project.

5.5.3 Random Forest:

For the project, we utilized a Random Forest classifier as our classification model to predict the project state based on the given features. The dataset was split into features (X) and the target variable (y), where the target variable represents the project state. The dataset was then divided into training and testing sets using the train_test_split function from scikit-learn. The Random Forest classifier was initialized without specifying any hyperparameters. We defined a parameter grid to search for the best combination of hyperparameters using the GridSearchCV class from scikit-learn. The hyperparameters we explored were 'n_estimators' (50, 100, or 200), 'max_depth' (3, 5, or None), 'min_samples_split' (2, 5, or 10), and 'min_samples_leaf' (1, 2, or 4).

Using 5-fold cross-validation, the GridSearchCV performed an exhaustive search over the hyperparameter combinations to identify the best parameters that maximize the model's performance. The best parameters found were 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, and 'n_estimators': 200. The best score achieved during the cross-validation was 0.9807, indicating a high level of accuracy.

The Random Forest classifier with the best parameters was then fitted to the training data using the fit() method of the GridSearchCV object. After training, predictions were made on the testing data using the predict() method, and the accuracy of the model was calculated by comparing the predicted values to the true values of the testing set. The resulting accuracy score was 0.9829, indicating that the Random Forest model achieved a high accuracy of 98.29% in predicting the project state.

Finally, the trained Random Forest classifier with the best parameters was added to the models_up list, indicating that it was one of the models evaluated in the project.

5.5.4 SVM:

For our project, we employed a Support Vector Machine (SVM) classifier to predict the project state based on the provided features. We used the dataset with the target variable 'state' and split it into training and testing sets using the train_test_split function from scikit-learn.We performed a hyperparameter tuning using the GridSearchCV class from scikit-learn to identify the optimal combination of hyperparameters. The hyperparameters we explored were 'kernel' (linear or radial basis function), 'C' (0.1, 1, or 10), and 'gamma' (0.1, 1, 'scale', or 'auto'). We utilized 5-fold cross-validation during the grid search to evaluate the performance of different parameter combinations.

The SVM classifier was initialized, and the GridSearchCV object was set up with the SVM classifier and the defined parameter grid. The GridSearchCV object was then fitted to the training data, which conducted an exhaustive search over the hyperparameter combinations to find the best parameters. The best parameters discovered were 'C': 10, 'gamma': 'auto', and 'kernel': 'rbf'. The corresponding best score achieved during cross-validation was 0.8813, suggesting a high level of accuracy. Using the best estimator obtained from GridSearchCV, we made predictions on the testing set. The classification report was generated, which includes precision, recall, and F1-score for each class (0 and 1). Additionally, the accuracy score was calculated, and it resulted in 0.8927, indicating a high level of accuracy in predicting the project state.

In conclusion, the SVM classifier with the optimized parameters demonstrated promising performance in our project. The best parameters, obtained through GridSearchCV, were 'C': 10, 'gamma': 'auto', and 'kernel': 'rbf'. The SVM model achieved an accuracy score of 0.8927, showcasing its effectiveness in predicting the project state based on the provided features.

5.5.5 Algorithm Comparison:

In order to compare the performance of different models, we employed K-fold cross-validation and evaluated their accuracy scores. We utilized a 10-fold cross-validation strategy, meaning the dataset was divided into 10 equal parts (folds). For each model, we iterated through the folds and performed cross-validation by training the model on 9 folds and testing it on the remaining fold. This process was repeated 10 times, with each fold serving as the testing set exactly once. We used the `cross_val_score` function from scikit-learn to obtain the accuracy scores for each iteration.

The models used for comparison were logistic regression, decision tree, random forest, and support vector machine (SVM). We stored the accuracy scores for each model in the `results` list and their corresponding names in the `names` list. For each model, we printed the average accuracy score across the 10 iterations (`cv_results.mean()`) and the standard deviation (`cv_results.std()`). These statistics provide insights into the consistency and performance of each model. To visualize the comparison, we created a boxplot using the `plt.boxplot` function from Matplotlib. The boxplot allows us to compare the distributions of accuracy scores for each model, providing a visual representation of their performance.

In conclusion, the boxplot comparison of the models' accuracy scores helps us understand their relative performance. It provides a comprehensive overview of how each model performs on average and the variability in their results. This analysis aids in the selection of the most suitable model for predicting the project state based on the provided features.

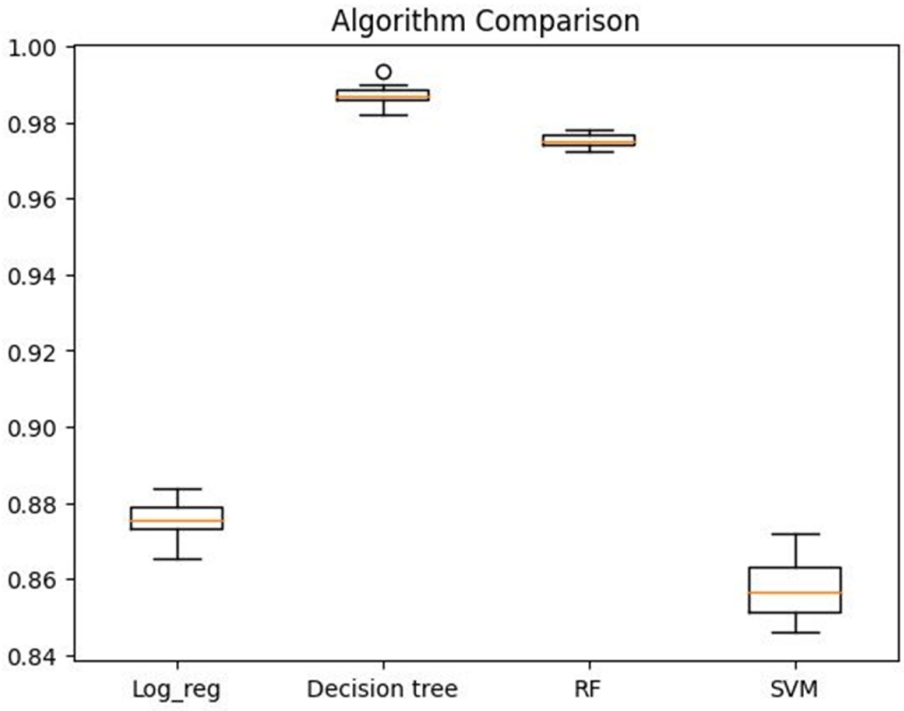

fig.5 Cross-Validation of models on Unbalanced data

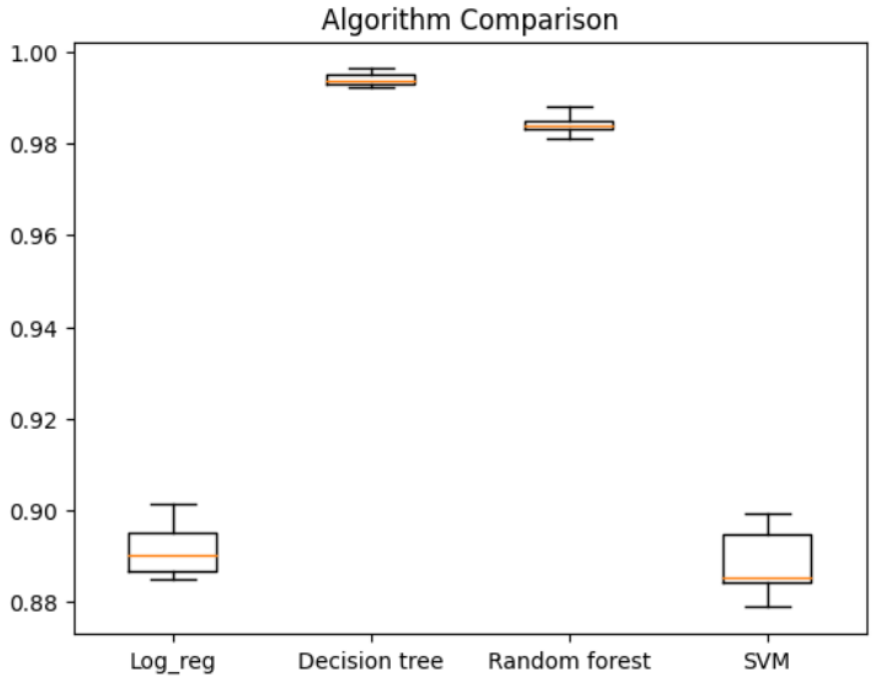

fig.6 Cross-Validation of models on Balanced data

5.6. Task Contribution:

Vikas Tadepu	Performed initial exploratory data analysis and trained the model on random forest as well as SVM algorithms. PCA was performed to transform the data into a lower-dimensional space. Assisted on the presentation slides.
Surya Teja Palle	Assisted with exploratory data analysis and worked on Logistic regression to train the model. Found the most suitable metric to evaluate the performance of the models and worked on the report.
Vineeth Reddy Thalasani	Assisted with exploratory data analysis and worked on training the model on Decision tree and created comparative analyses for all the algorithms. Also worked on the presentation slides and the report

6. Conclusion and Results

We evaluated multiple machine learning models to predict the success of Kickstarter projects. After training and testing these models on our dataset, we obtained the following accuracy scores:

1. Logistic Regression: Achieving an accuracy of 0.8913, logistic regression demonstrated a strong performance in predicting project success. This model utilizes a linear decision boundary and probabilistic framework to classify projects.
2. Decision Trees: Outperforming other models, decision trees exhibited an impressive accuracy of 0.9923. Decision trees partition the feature space based on a series of if-else conditions, enabling them to capture complex patterns and relationships within the data.
3. Random Forest: With an accuracy of 0.9801, random forest demonstrated robust predictive capabilities. It combines multiple decision trees through ensemble learning, resulting in improved generalization and reduced overfitting.
4. Support Vector Machines (SVM): SVM achieved an accuracy of 0.8927, making it a competitive model for Kickstarter project success prediction. SVM constructs a hyperplane that maximally separates the classes in the feature space, leveraging both linear and nonlinear kernels.

Model		Accuracy
Logistic Regression	Imbalanced Data	0.8808
	Up-sampled Data	0.8914
	Down-sampled Data	0.8862
Decision Trees	Imbalanced Data	0.9852
	Up-sampled Data	0.9923
	Down-sampled Data	0.9884
Random Forest	Imbalanced Data	0.9728
	Up-sampled Data	0.9801
	Down-sampled Data	0.9630
SVM	Imbalanced Data	0.8913
	Up-sampled Data	0.9247
	Down-sampled Data	0.9028

Table.1 Comparing the accuracies of each model

Overall, We successfully predicted the kickstarter project success using machine learning models. Exploratory data analysis gave us valuable insights about the dataset and the correlation of the features with respect to the target variable. The Decision Tree Algorithm on unsampled data worked best for us and feature selection and performing Principal Component Analysis was proven helpful.

Appendix

a. Source Code: https://github.com/vineethreddythalasani/CMPE-257-Final-Project

b. References:

1. Borooah, S. (2017). Multicollinearity: Effects, Importance and Solutions. *International Journal of Business Research and Management*, 8(1), 1-12.
2. Liu, W., Li, X., & Li, X. (2020). Using machine learning methods to predict the success of crowdfunding projects: A case study of Kickstarter. *Sustainability*, 12(7), 2715.
3. Li, Y., Rakesh, V., & Reddy, C. K. (2016). Project success prediction in crowdfunding environments. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (pp. 247-256). ACM.
4. Xu, B., Liu, S., Wang, F., & Zhang, Y. (2018). KickPredict: Predicting Kickstarter project success with natural language processing and neural networks. *arXiv preprint arXiv:1803.04311*.
5. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.
6. Ahmed, T., Tyagi, V., & Kaur, R. (2017). Predicting crowdfunding success with optimally weighted random forests. In *2017 13th International Conference on Information Technology and Applications (ICITA)* (pp. 769-775). IEEE.
7. Belleflamme, P., Lambert, T., & Schwienbacher, A. (2015). Crowdfunding: Tapping the right crowd. *Journal of Business Venturing*, 30(6), 1049-1067.
8. Mollick, E. (2014). The dynamics of crowdfunding: An empirical exploration of factors affecting success and failure. *Journal of Business Venturing*, 29(1), 1-16.