Varsha Swaminathan

Problem Statement: Predicting Kickstarter project success

Understanding the business problem:

What is Kickstarter?

It is an online fundraising platform where creators post their projects and ideas along with specs for their finished product, potential risks and progress updates, then other people can donate money to these projects.

So, it is basically a crowdfunding platform where the creators can get donations from a larger number of people at once which reduces the amount of money each donor has to put in.

It uses an all or nothing funding strategy. So, a creator sets a monetary goal and a deadline, then supporters can pledge money towards the completion of the projects. If the project does not meet its funding goal by the set deadline, then the creator receives no money and the supporters will not be charged for their pledges.

Kickstarter has different categories for project ideas and creators stay accountable to their supporters. Many projects on Kickstarter have incentive for people who make donations above a certain amount.

For our problem statement, we are trying to predict if a proposal will succeed or fail based on information like project category, fundraising goal and short description of the proposed product. We attempt to use information that's available at the time of product launch to predict success or failure.

Gathering relevant data:

We collate data for our problem statement from webrobots.io. The website gathers data through web crawling each month from the Kickstarter site in a CSV/JSON format. The dataset has various attributes like the funding goal of the creators project, project categories, duration, a short project blurb, number of backers for the project, the creator information like their name, external links to their social media, biographies, previous Kickstarter activities, etc. However, we attempt to use only the information available at the time of project launch.

For our dataset, we use information available from April 2019 upto April 2021. This data could contain duplicate rows, we filter out these values on the project name, blurb, launch date and deadline and keep only the unique entries relevant for our purpose. We will now have around 221,248 completed Kickstarter campaign information. We do a 70-30 train test split on this data.

Our aim is to keep the attributes available at the time of launch:

1. Project target funding goal in USD
2. Launch date
3. Deadline of the project
4. Project blurb
5. Project category
6. Location of the creator

Information like the number of backers, funding received thus far and time remaining is not available at the time of launch and hence excluded from our project analysis.

Data preparation and EDA: (which also includes feature engineering and feature extraction)

Categorical Encoding:

As discussed above, the attributes we consider have a handful of categorical variables. Hence, we need to do categorical encoding. We use the mean encoding strategy for this.

- *So, for each unique value of a categorical feature, we replace it based on the ratio of occurrence of the positive class (Success) in the target variable.*

Here, mean encoding represents the probability of our target variable, conditional on each value of the feature, making our feature more representative of the target variable.

Here, we also need to keep in mind that using target variable to encode categories may in turn cause the variable to become biased. Hence, to prevent this we use cross validation. Essentially this prevents data leakage and so it prevents overfitting the model and biasing the feature.

Steps for encoding categories:

If a categorical column has a unique value with greater than 1000 projects, we perform a conditional target mean encoding explained above in blue. If the unique values have less than 1000 projects then we encode with an unconditional overall target mean. Here, to avoid data leakage, we use an 'out of fold' mean estimate. So, we use the average k fold method, where for each fold the average of the remaining k-1 folds mean encoding value is used to replace the unique category of a categorical variable.

- For each categorical feature, we also introduce an out of fold mean encoding using project funding goal. We also consider the difference between the creator projects funding goal and this obtained mean value of goal.
- We also do a one hot encoding of the project category and parent category features. But we don't do one hot encoding for all categorical variables as we want to preserve the cardinality of our feature space, i.e., number of features in our mathematical space. Because one hot encoding variables like location which have over 13000 unique values will result in too many features and additional dimensions to our mathematical feature space, which might also lead to overfitting hence we try to avoid this.

Text Encoding:

We have a feature which gives a small description(blurb) for each project.

We first pre-process this blurb to remove redundant words or stop words and non-alphanumeric characters, we then use various encoding methods to create new meaningful features from these blurbs:

1. Character length of this blurb is used as a feature
2. Unigram count matrices which uses tf-idf transformation:

TF captures how important a word is to a project blurb, without looking at other project blurbs in the dataset. IDF tells us if a word can be used to distinguish projects. For example, a common word in all the blurbs will have IDF close to 0 as it is not very informative.

3. Latent Dirichlet Allocation (LDA):

It allows unsupervised clustering of words into predefined number of clusters (Topics). LDA assumes that blurbs are composed of words that help determine the topics and maps burbs to a list of topics by assigning each word in the blurb to different topics. The assignment is in terms of conditional probability estimates. the value in each cell indicates the probability of a word belonging to topic. It is important to note that LDA ignores the order of occurrence of words and the syntactic information. It treats documents just as a collection of words or a bag of words.

4. Word Embeddings (Word2Vec):

Based on the idea that words occurring together are mostly related and so it captures the similarity of semantic meanings between words.

5. Sentiment analysis to provide a positivity rating of the blurb. We use Stanford CoreNLP to to analyse the blurb and get a sentiment score between 0 indicating least positive and 4 indicating most positive. The score is then used as a feature to confirm whether there exists a positive correlation between more positively framed campaigns and success rates.
   Here we assign score for each sentence in a blurb independently and then take the average sentiment score for the overall blurb.

Now we have all our features and we are ready for model building.

Model Building and Deployment:

Our task is a binary classification problem. To predict success (1) or failure (0) of a Kickstarter project in meeting its fundraising goal.

We have chosen our threshold as 0.5 for all models with continuous output values like linear or logistic regression.

The models we will be using are:

1. Multinomial Naïve Bayes (MNB):

MNB is a supervised learning algorithm which assumes to have feature vector with each feature representing the number of times it appears or its frequency, which is exactly the kind of features we have here. It considers that the samples are drawn from a multinomial distribution and are conditionally independent given the outcome.

2. Linear Probability Models
   - Ordinary Least squares (OLS): the least squares error function/cost function is minimized using the gradient descent algorithm.
   - Lasso and ridge regression: The are similar to OLS but with added penalties in the cost function where lasso takes the absolute sum of weights but ridge takes the sum of squares of the weights. As we increase the lambda parameter it causes the weights/coefficients of redundant features to tend to zero hence helping reduce dimensionality of the feature space.

3. Logistic Regression: Logistic regression uses a sigmoid function as here we do not have a continuous target variable but a two-class categorical target, so we need an S curve using the sigmoid function to model this.

4. Support Vector Machines (SVM): SVM classifier draws a line or a plane between two classes where the plane stays far away from the closest samples of the two classes, this is the maximum margin. All data points that fall on one side of the plane are labelled as one class (positive one or success) and the data points that fall on the other side are the second class (negative one or failure).

5. Random Forest: It uses bagging method to combine individual predictions of many decision trees using a method called voting to obtain a majority vote as final classification. It uses a sampling with replacement method and only a subset of the original features are picked up in each split. This prevents the individual trees from being similar, thereby allowing it to explore more of the overall feature space and ensuring that the subsequent aggregation of the trees is more robust.

6. Gradient Boosting: It is an ensemble method using boosting where iteratively it fits the weak learner on the residual at each step, attempting to correct the errors made by the previous step. The base model is the log odds of the target variable, then we calculate the residuals and use them to calculate the next tree. Learning Rate is used to scale the contribution from the new tree.
   Old Tree + Learning Rate * New Tree
   We can now calculate new log(odds) prediction and hence a new probability. This process repeats until we have made the maximum number of trees specified or the residuals get super small.

7. Dense Neural Network: They allow for learning of complex non-linear models through a series of connected layers. Each neuron in the dense layer receives input from all neurons of its previous layer. In forward propagation, each neuron has an activation a and each neuron is connected to a new neuron that has a weight w, now multiply activations by weights and get a single neuron in the next layer.
   $w_1a_1+w_2a_2+...+w_na_n$=new neuron
   In backward propagation, we start from the output layer and propagate backwards, updating weights and biases for each layer. Adjust the weights and biases throughout the network, so that we get the desired output in the output layer. We use the Adam optimizer with binary log loss function.

Model Evaluation Metrices:

We use the overall accuracy, precision, recall and F1 score to evaluate our models. We do not have an unbalanced target variable and for our problem statement False positives do not have a grave impact so we consider accuracy to be an important indicator of performance as it places equal importance on the true positives and true negatives. As F1 score penalizes the false predictions more and places less value on true negatives as true negatives are not used in the formulae for precision and recall and F1 is the harmonic mean of precision and recall, F1 and accuracy are our important metrices.

Varsha Swaminathan

Text models:

1. Naive Bayes classifier uses unigram occurrences and applies TF-IDF transformation. Fitting this to the model, we obtain probabilistic predictions for the success of each project. We then use the cut off of 0.5 we chose earlier to classify the samples as success or failure based on these probabilities.
2. We use Stanford CoreNLP to annotate the blurbs, requesting a sentiment value between 0 and 4 for each blurb. Here we assign score for each sentence in a blurb independently and then take the average sentiment score for the overall blurb score. We then fit a logistic regression between the sentiment score and project outcome.
3. For an LDA model, we drop all tokens which occur in fewer than 10 blurbs or more than 35% of blurbs. We specify 20 latent topics with identical priors, and this is on the order of Kickstarter's parent project categories (15).
4. For word embeddings, we use Google's pretrained W2V model with 300 dimensions from a 100 billion word Google News corpus. For each blurb, we compute the average of the associated W2V vectors. Fit these features to a logistic regression model.

These text models individually don't perform as well but when combined with the external project metadata their performance increases.