KICKSTARTER PROJECT

Version:1.0

Manjeel Baral
Data Science Fellow
General Assembly

Data Science Work Flow

Define Problem

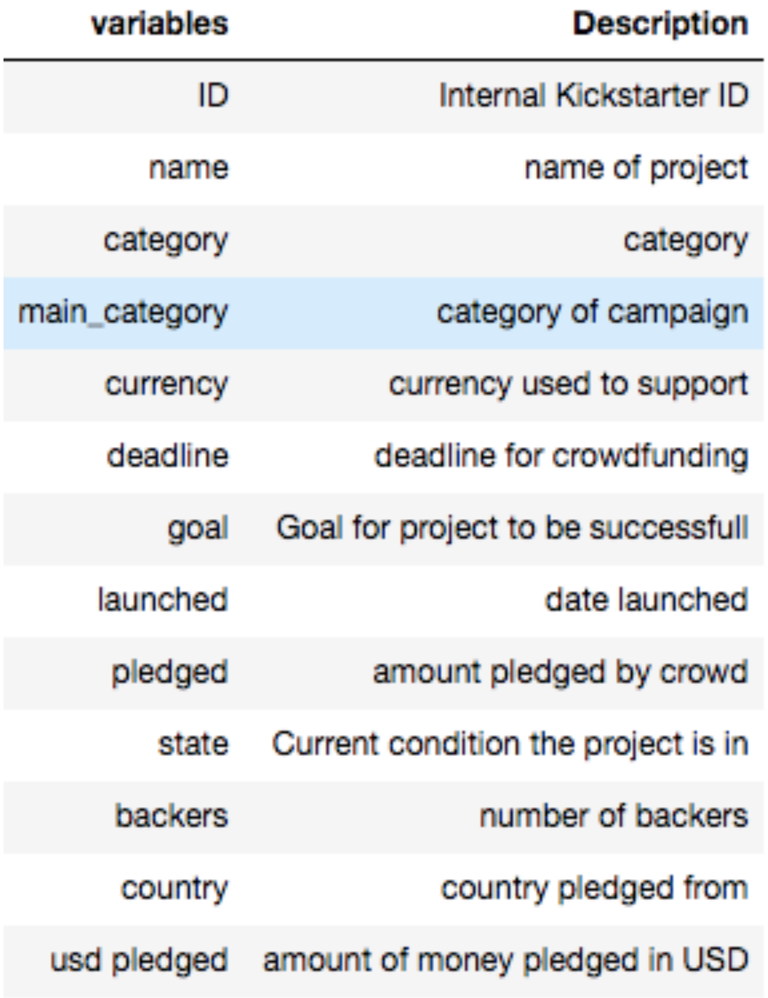
The primary goal of my capstone project is to predict whether the project is successful or failed given the explanatory variables.

The Kickstarter is an American public-benefit corporation that supports and hosts a global crowdfunding platform to harbor and encourage creative projects and startups specifically for creative projects in the following categories: Art, Comics, Crafts, Dance, Design, Fashion, Film & Video, Food, Games, Journalism, Music, Photography, Publishing, Technology, and Theater. The members of the audience who decide on funding the project is called backers and the amount they contribute is called the ‘pledged amount’. If project does not gather funds (pledged amount) that is equal to the goal than that project is failed and therefore backers will not be charged.

Gather Data

I obtained Kickstarter project data from Kaggle with 12 explanatory variables. Description of obtained variables are described below:

- ID
- Name
- Category
- Main category
- Currency
- Deadline
- Goal
- Launched
- Pledged
- State
- Backers
- Country
- USD pledged



variables	Description
ID	Internal Kickstarter ID
name	name of project
category	category
main_category	category of campaign
currency	currency used to support
deadline	deadline for crowdfunding
goal	Goal for project to be successfull
launched	date launched
pledged	amount pledged by crowd
state	Current condition the project is in
backers	number of backers
country	country pledged from
usd pledged	amount of money pledged in USD

Explore Data

This is one of the most important steps in Data Science workflow as it depends on the predictability and interpretability of the model. Some of the important steps I have taken while exploring the dataset is pointed out below:

- Read the CSV in the Jupyter notebook and checked the first five rows followed by checking the datatypes, info, shape and null values
- Drop the unnecessary columns
- As we are only concerned with successful and the failed project, I have filtered the data frame with successful and failed projects.
- Data frame has two variables deadline and launched which has date and time and as date and time can be important features I have spllited the column to sperate date and time from deadline and launched column and made new column.
- Since time and date was separated to make new columns deadline and launched columns was dropped.
- Since we have now launch date and deadline date, I have converted these two variables to datetime object to get a new feature project duration which could be another important variable in determining whether project was successful or failure.

- Again, from date launch and deadline date I have splitted year, month and day to make extra new features and dropped date launch and deadline date column and made extra 6 columns year launch, month launch, day launch, deadline year, deadline month, deadline day.
- Time launch and deadline time was on 24-hour format I have converted it to 12-hour format to make a new feature and dropped the launch and deadline time column
- After taking all the steps the main step is to convert the datatype.
- Visualized different attributes with total number of projects (successful/failed)

Pre-processing step before modeling

- First step was to separate data frame by categorical features and numerical features
- We can't fit the model with text data we have to convert the Categorical data to numerical features therefore I have used Label Encoder to convert categorical features into numerical features. I could not use One Hot Encoder due to memory error
- Merged back the Data Frame
- Splitted the data into training set and testing set

Model the Data

Since the goal of my project is model predictability, I have leveraged the concept of ensemble methods and used Gradient Boosting classifier to fit the data.

Boosting is a sequential technique which works on the principle of **ensemble**. It combines a set of **weak learners** and delivers improved prediction accuracy. At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differential loss function.

Gradient boosting classifier algorithm was used and was fitted the training set and evaluated on testing set.
The train Accuracy score was 0.93
The test score Accuracy was 0.929
Cross validated score was 0.93 (mean)

Evaluate the Model

After building the classification model we can evaluate model on following basis:

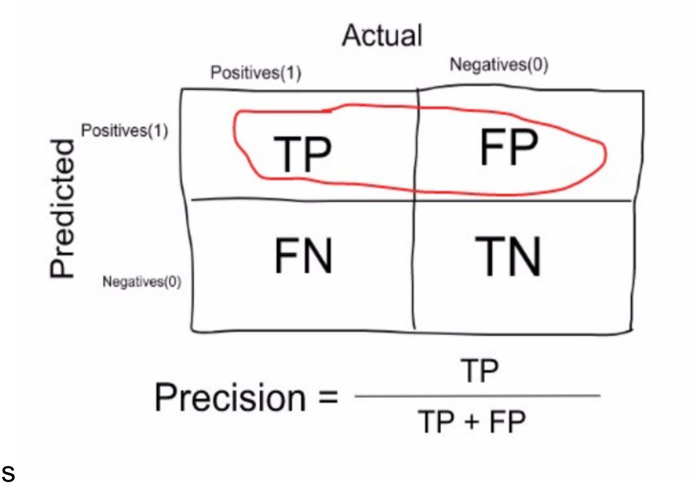
- Model truly predicts positive class: True Positive
- Model truly predicts negative class: True Negative
- Model falsely predicts positive class: False Positive
- Model falsely predicts negative class: False Negative

Scikit-learn makes more easier with confusion matrix which helps us to understand what instances of positive classes classified as negative class and what instances of negative class classified as the positive class.

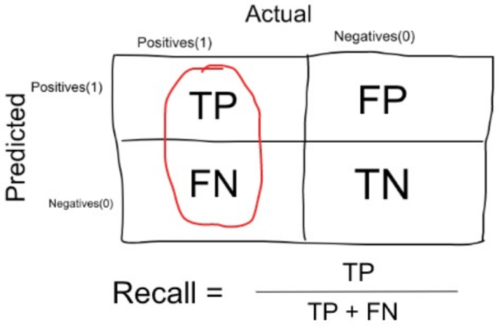
- Model truly predicts negative class, True Negatives: 39193
- Model falsely predicts positive class, False Positives: 2836
- Model falsely predicts negative class, False Negatives: 2123
- Model falsely predicts positive class, True Positives: 26121

  False positive and False Negative are called as Type 1 and Type 2 errors respectively and its effect depends on your business objective. For instance, if you are trying to predict whether or not the transaction was fraud and the model did predict the real transaction (False Positive) as fraud it's not a big problem but if actual transaction was fraud but model did not predict (False Negative) than it's a big problem. Similarly, False positive can be a big issue too if your boss sends you email but model predicts the actual email as spam (False positive).

Further, Scikit-learn has classification report which provides precision and recall (sensitivity) score. Precision in the classification model is the accuracy of positive predictions. Example: What proportion of successful project that was actually successful was predicted as positive class (score: 92%) If precision is what we are trying to increase, we have to decrease our False positives



Similarly, Recall Also called the true positive rate is the ratio of positive instances that was correctly detected by the classifier. In our classification project the ratio of actual failed project that was correctly classified as a failed project was 90 %. To increase our sensitivity or Recall we have to minimize our false negatives and it completely depends on business objective



	Positives(1)	Negatives(0)
Positives(1)	TP	FP
Negatives(0)	FN	TN

$$\text{Recall} = \frac{TP}{TP + FN}$$

Specificity also called as the true negative rate is the ratio of negative instances that was correctly detected by the classifier. To increase the specificity, we need to decrease the false positive and it completely depends on business objective.

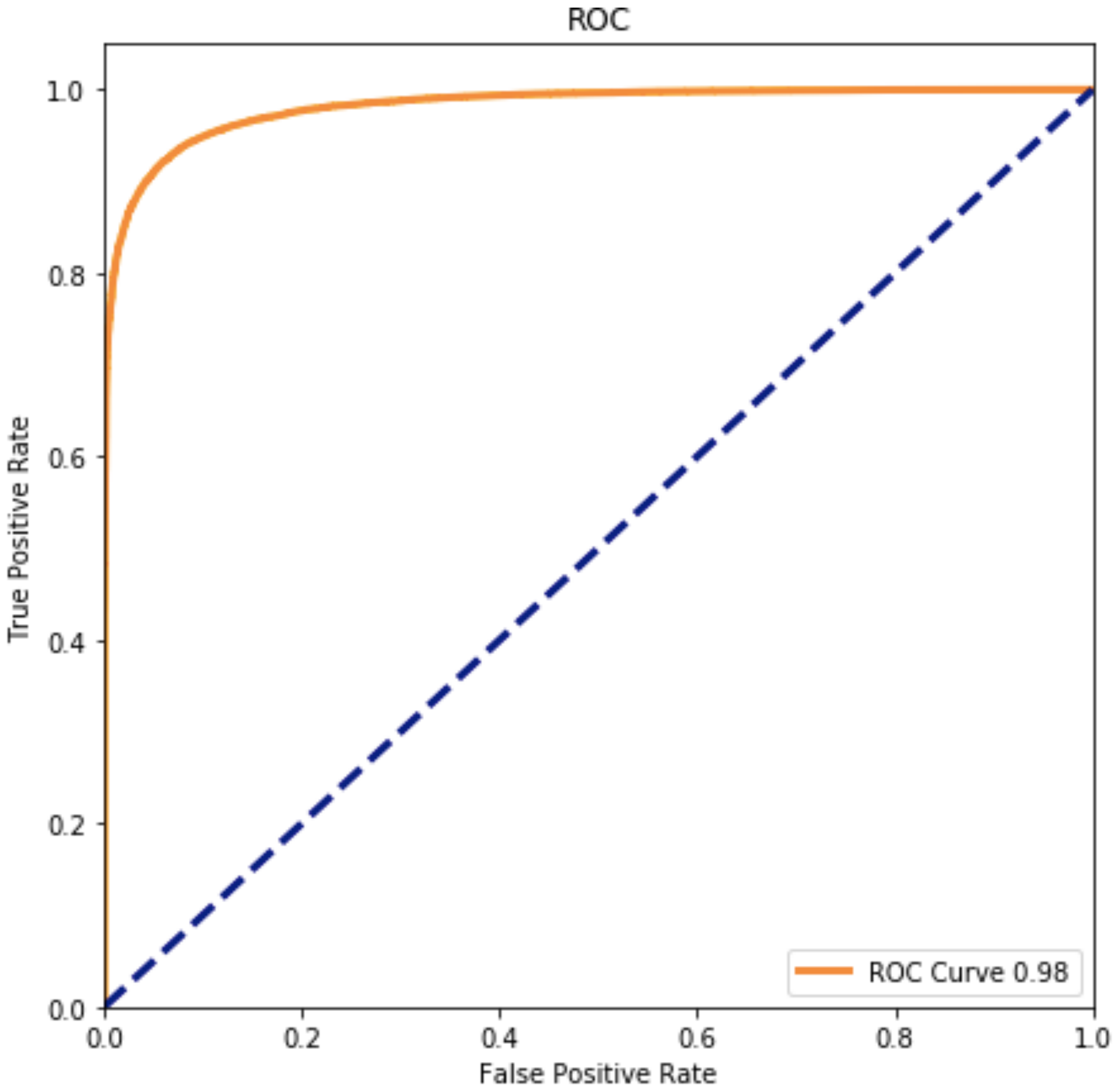
AUC- ROC Curve

We can use the idea of **sensitivity** and **specificity** in a combined metric we typically plot that represents both of these stats combined into one curved line, called The **Receiver Operator Characteristic**.

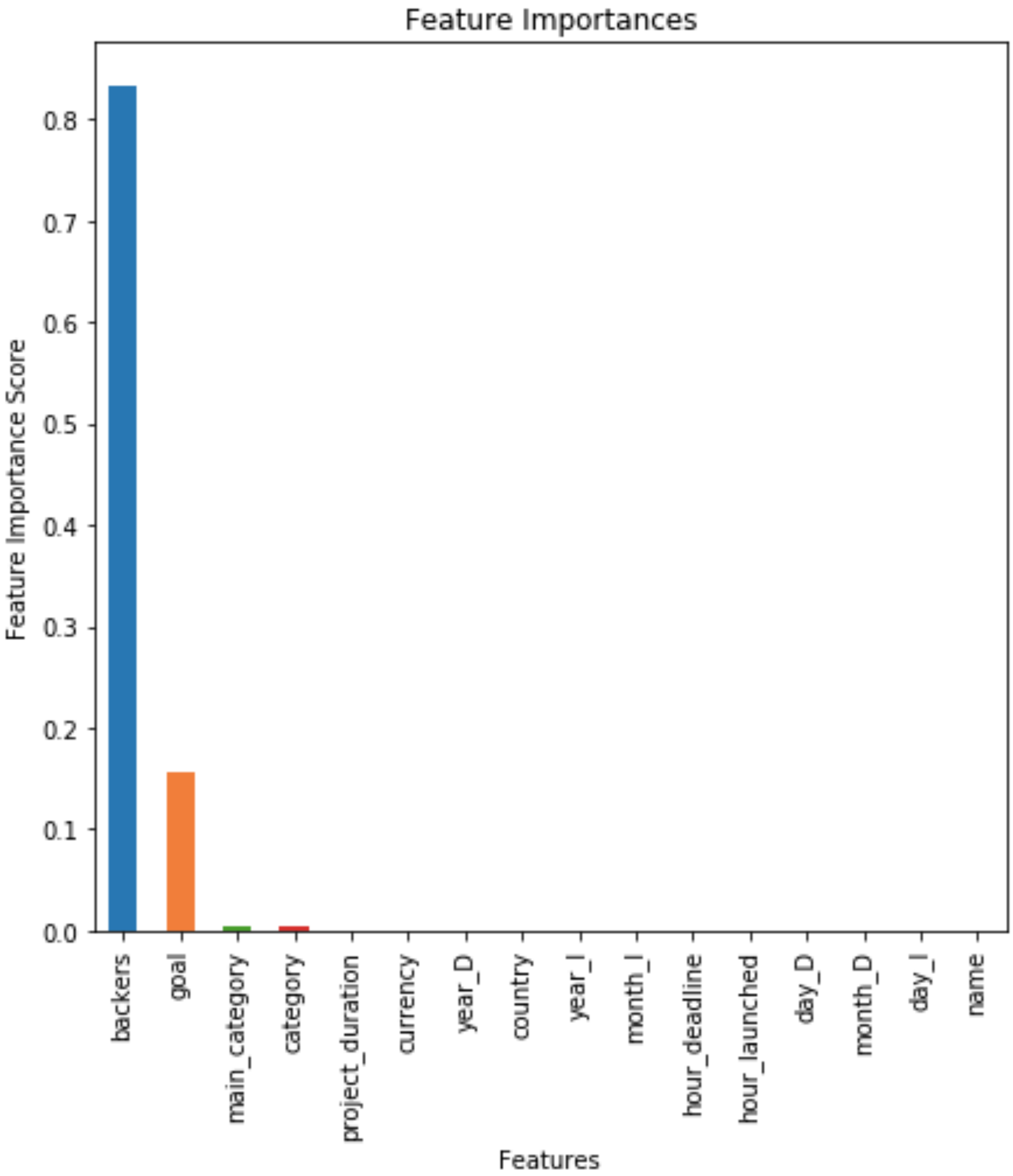
We generate the ROC by plotting the sensitivity and specificity as we move our “classification threshold” from 0 to 1. The ROC gives us a sense about the tradeoff between sensitivity and specificity. Also, the closer the curve is the upper left, the better overall accuracy of our model is. The area under the curve is also referred to as **AUC**. The acronym **AUC-ROC** refers to the “Area Under the Receiver Operating Characteristic curve.”

We generate one ROC curve per model. The ROC curve is generated by varying our threshold from 0 to 1. This doesn't actually change the threshold or our original predictions, but it does tell what our tradeoff between *sensitivity* and *specificity*.

AUC-ROC curve generated by model with AUC-ROC score of 0.98



Important features detected by Model



Answer the problem

Algorithm has given the most important features as backers and it makes sense as Kickstarter has adopted All-or-nothing funding meaning that no one will be charged for a pledge towards a project unless it reaches its funding goal and to make project successful it needs backers to support and share about the project (word-of-mouth). So, the project being pledged is completely dependent number of backers a project can get. The one who starts the project should give maximum time in making project more creative, being very sincere to the question asked by the potential backers as backers that support a project on Kickstarter get an inside look at the creative process.