**Machine Learning**

I split the features in two section, one for applying NLP on campaign title and the other set holds all the relevant features derived in earlier analysis.

1. **NLP**

Name is the only feature with text in this data set. However, the contents are not purely numeric or alphanumeric. There is a lot of special characters and symbols, along with numbers. Some examples are, 3X3, 000, #007, etc.

Prepare data

Name text could be vectorize in few different ways. I used word\_tokenize initially to split the text into list of words. This was not the best method as it considered any string between two white spaces as a word. This was corrected by first removing any special character, numbers and punctuation. However, this could be very easily done by using regular expressions as token pattern parameter in any vectorizer.

Vectorizer

I used CountVector (CV) and TfIdfVector (TV) as vectorizers with token\_pattern = '[A-Za-z0-9]+'. TV is not the best option and this was confirmed by reduced cross validation score. TV does not consider words which are commonly occur in all observations. While this may be a better choice in analysis where all the observations belong to similar context, for example fiction books. In this analysis, campaigns are registered by different countries and dialects, humor, emotion, expression is not presented in the same manner as US. This when combined with bi-gram analysis produced better results. CV worked better as it accounted for all words.

Classification model

Naive bayes produced better results than logistic regression or random forest. Cross validation score of 0.645 with bi-gram and parameter tuning with alpha = 1. This is a very much out of the box model for simple analysis to provide structure and basic understanding in capstone 1. Accuracy on training data is 0.92 and on test data is 0.65. This screams overfitting.

Dimensionality reduction

PCA did not work on this since the input matrix is very sparse. I used TruncatedSVD to produce 2 features on a scatter plot. The plot does not show any clear different between two classes, successful and successful. This is pretty obvious as confirmed by poor model accuracy.

1. **Classification model**

I converted few date features into month and day. Here is the list of independent variables.

finish\_month

finish\_day

start\_month

category

main\_category

country

year

duration

goal

Prepare data

First three features are used to assess if there is any seasonality factor which could influence the campaign success. Country, category and main\_category are non number variables. I used them along with year as categorical variables. Year value ranges from 2009 to 2017. This is numeric but it would not make sense to have 2016.5 as a year. Thus, lets treat each year as a category. Duration and goal are numeric variable and the most correlated with result, as seen from earlier analysis.

Pipeline

I created two pipelines, each for numerical and categorical variables. For numeric, steps include simple imputer and standard scaler. For categorical, steps include simple imputer and one hot encoder. A third pipeline is created as column transformer.

I. numeric\_transformer = Pipeline(steps=[('imputer', SimpleImputer(strategy='median')),('scaler', StandardScaler())])  
  
II. categorical\_transformer = Pipeline(steps=[('imputer', SimpleImputer(strategy='constant', fill\_value='missing')),  
 ('onehot', OneHotEncoder(handle\_unknown='ignore'))])  
  
III. preprocessor = ColumnTransformer(transformers=[('num', numeric\_transformer, numeric\_features),  
 ('cat', categorical\_transformer, categorical\_features)])

This preprocessor is then fed in the final pipeline with classifier model. Gradient Boosting produced better results than logistic regression and random forest. Test score of 0.682.