Big Data Group Project – Phase 2

# Overview

Phase 2 of the project is the development of a cleansing and reshaping pipeline for our input data. Our dataset, as outlined in Phase 1, is a list of Kickstarter campaigns and attached information such as campaign name, category, launch and deadline dates, number of backers, fundraising goals, and amount pledged. Our dataset input consists of two files – ‘ks-projects-201612’ and ‘ks-projects-201801’. The process undertaken for this Phase was as follows:

Step 1) Confronting both datasets. The two datasets were explored and wrangled into one dataset. The final decision made was to use the 2018 dataset exclusively for input into the pipeline. See section for further detail.

Step 2) Exploratory Data Analysis on the 2018 dataset. This was undertaken to understand the dataset and justify any changes required. This analysis informs the transformation steps undertaken for the remainder of the process. The remaining steps were undertaken in order to clean data and produce better features as inputs into machine learning models for the following Phase.

Step 3) Adding feature length for Project Name Length. Campaign names were converted into a quantitative variable by creating a variable with word counts of the campaign name, binned into quartiles.

Step 4) Manipulate dates and time. Month. year, and day duration values were extracted from the date fields for use as features.  
  
Step 5) Encode continuous variables. For ease of use as a feature, continuous variables in the dataset will be binned into discrete values from 0 to 9.

Step 6) Encoding categorical variables. For use as features within a logistic regression model, categorical variables were converted into dummy variables with a binary value of 0 or 1. This was done for the Category, Currency and Country. The ‘state’ of each project was also encoded to a 0 or 1 value, indicating if the project was ‘Successful’ (1) or not (0, encoded to all other categories). ‘Live’ (ongoing as at capture) projects are dropped.

Step 7) Filter Currency and Country Variables. Currency and Country variables are checked against an approved list of Kickstarter Currencies/Countries, and dropped if they don’t match.

Steps 3-7 were combined into the final cleansing and transformation pipeline.

The outputs for this project are:

1. This report
2. The pipeline of transformations - ‘pipeline.ipynb’. This pipeline was collated from individual team members’ work on each segment of the pipeline, as outlined in ‘Team Members and Contribution’ section below.
3. Initial EDA – ‘Preliminary\_Confronting\_Datasets.ipynb’. This EDA justifies the decision made to use the 2018 data as the only input into the pipeline.
4. An ‘Appendix’ folder. This folder contains all EDA done by individual group members on the 2018 dataset. Portions of this EDA are included in the below sections, and it informed the creation of steps 3-7 above which constituted the transformation pipeline.

# Team Members and Contribution

The team members, their roles and responsibilities for this Phase of the project are summarised below.

* **Alec David Vukovich**
  + EDA, Feature Name Length for Project Names, and collation of final pipeline.
* **Darragh Brendan Ruddy**
  + EDA, encoding of continuous variables, and filtering currency and country variables.
* **Jemal Mohammed-Nour**
  + EDA and manipulation of datetime variables
* **Matija Zivkovic**
  + Team leader for this Phase. EDA, confrontation of the 2016 and 2018 datasets, and compiling of this report.
* **Myrnelle Jover**
  + EDA and the encoding of categorical variables.

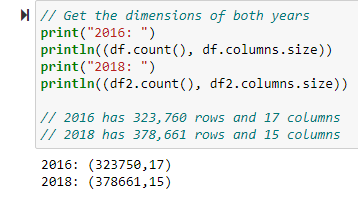
# Step 1) Confronting both datasets

For accompanying code, please see the ‘Preliminary\_Confronting\_Datasets’ notebook included with the submission.

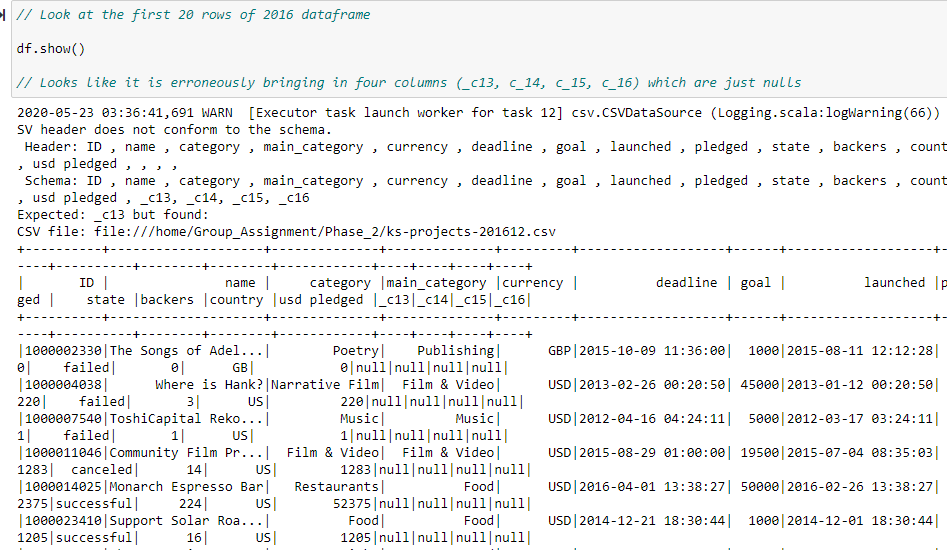
The Kickstarter data is provided as two datasets in .csv format: ‘ks-projects-201612’ and ‘ks-projects-201801’. Preliminary EDA was done to explore the datasets, with the intention of concatenating both datasets into a singular dataset.

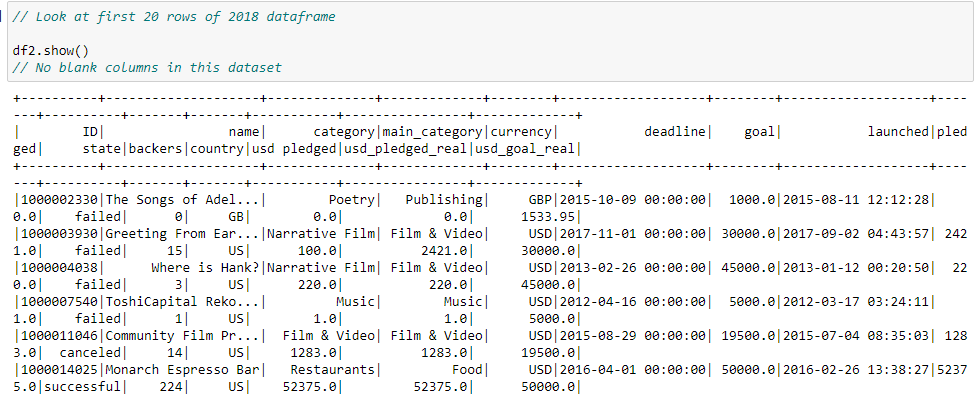
Upon investigation, it became apparent that the 2018 dataset encapsulated the 2016 dataset, and therefore only the 2018 dataset was required as input into the pipeline. Key EDA and wrangling steps are detailed below.

Get dimensions of both datasets.

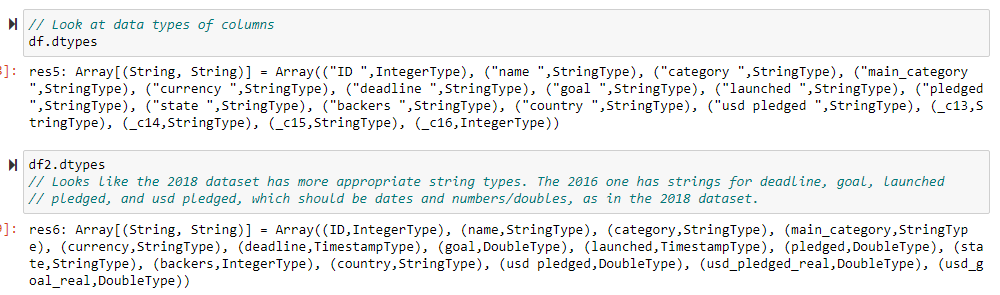


Show first rows of both datasets to get an idea of variables:



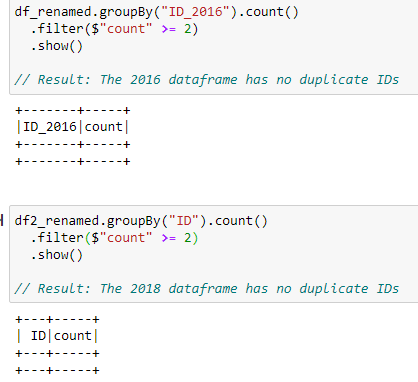


Columns and datatype listing for both datasets:

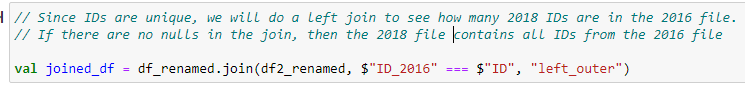


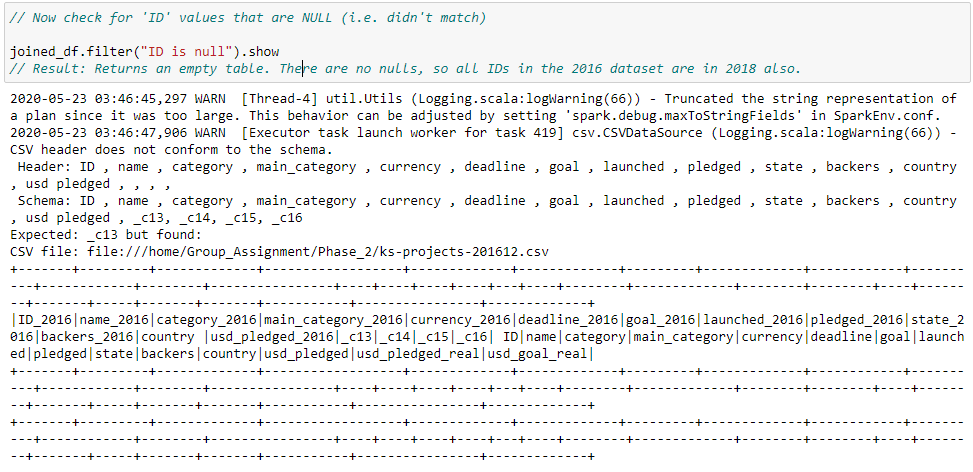
The 2018 dataset is better quality than the 2016 dataset. The 2016 dataset has 4 extraneous variables (\_c13, \_c14, c\_15, c\_16) which all had null values. Furthermore, all variable data types were stored as Strings in the 2016 dataset, whereas the 2018 dataset has data types appropriate to the variables. I.e. ‘usd\_pledged’ and ‘goal’ amounts are of DoubleType, deadline and launched are TimeStamp values, and backers is an integer indicating a number of people.

The ID value is confirmed to be unique in both datasets.

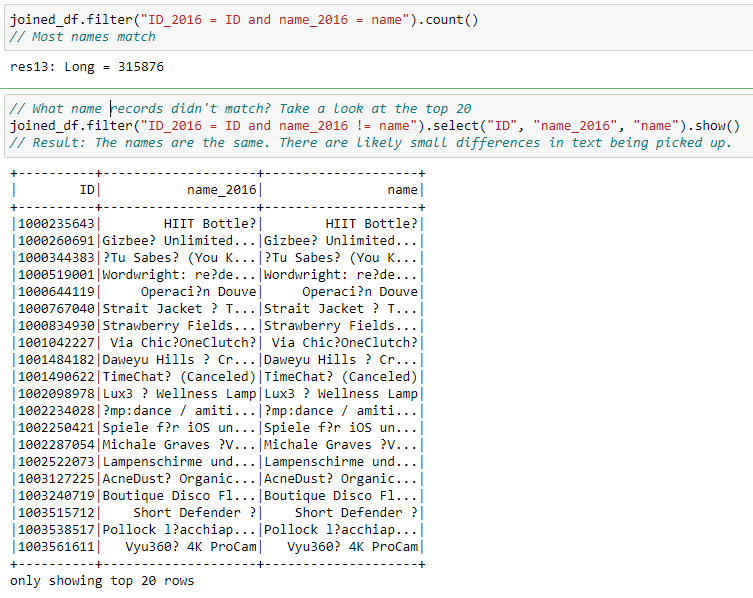


A left join was done, using ID as the key, to determine how many values in the 2016 dataset were in 2018. An absence of any nulls indicated that all IDs from 2016 were in 2018 also.

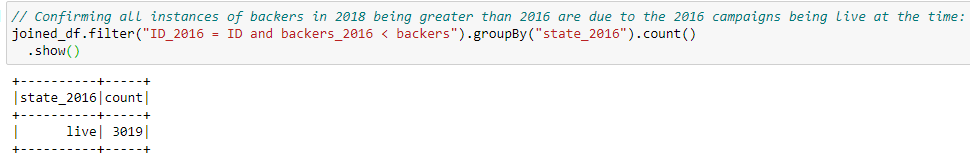


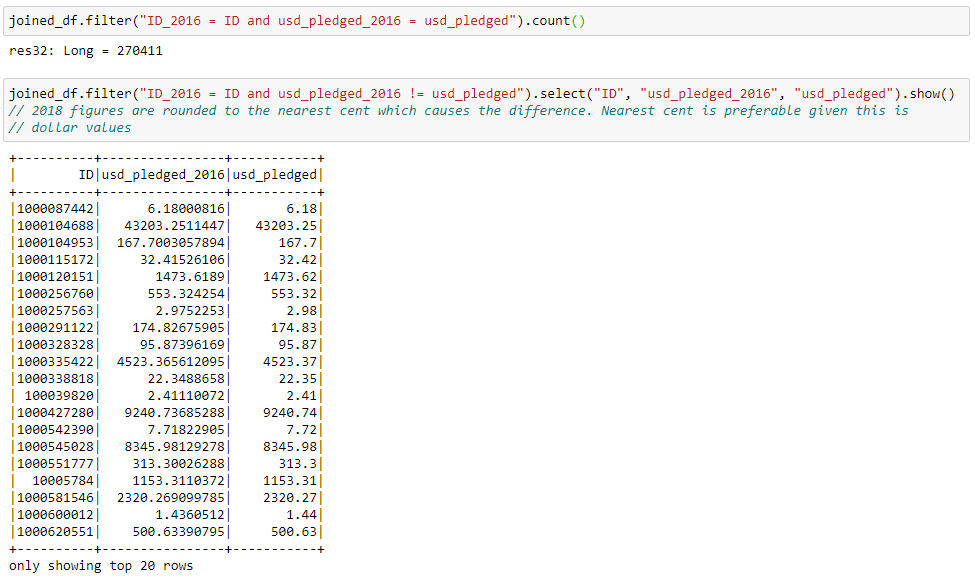


Counts were made of rows in the 2016 dataset vs 2018 with the same ID but different values for other variables. On looking at these variables it can be seen that they’re the same across datasets and had minor formatting differences. The below shows that 315,876 of 2016 names were the same as 2018 for the same ID. Those deemed different were actually the same, on inspection.



The only legitimate variable difference in 2016 vs 2018 data was ‘Backers’, but this was because 2018 backer counts were greater for Kickstarter campaigns that were still ‘live’ in 2016. This makes sense and demonstrates that 2018 includes updated data.

‘Usd\_pledged’ was different in 2016. However, this was because the 2018 data rounded the numbers to the nearest two decimal places. This is more appropriate, given that these are dollar and cent values:



We can conclude that the 2018 dataset includes all values from the 2016 dataset, and is also superior. Justifications:

1) It has all the IDs that the 2016 dataset has.

2) All the columns are assigned the correct data type when imported in 2018, but not for 2016.

3) 2018 has the same column values as 2016 for those records that are shared

4) It has updated 'backer' counts for those campaigns that were 'live' in 2016,

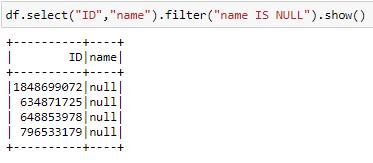
5) It has correctly assigned decimals, rounded to two places, for the usd\_pledged variable

Therefore, we will use the 2018 dataset exclusively as an input into the cleaning pipeline.

# Step 2) Exploratory Data Analysis on the 2018 dataset

Before doing any transformations on the dataset we need to explore the data to detect outliers, inconsistency or null values.

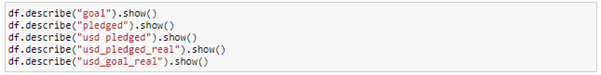
In the name column there were 4 null values.

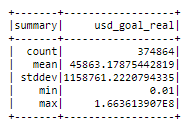
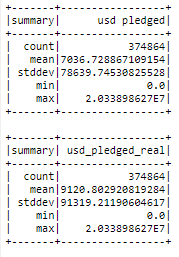
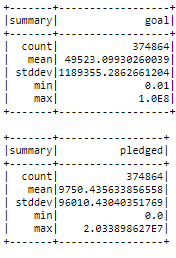


Exploring the continuous pledged and goal columns:



There are no null values in the continuous columns.

We now look at summary stats for the continuous variables:  
 



We can see there are no null values and the min and max values seem reasonable. The min is non-negative for all variables, which is good. For the highest pledged value, checking the dataframe for the row entry shows that the project name was Pebble Time:

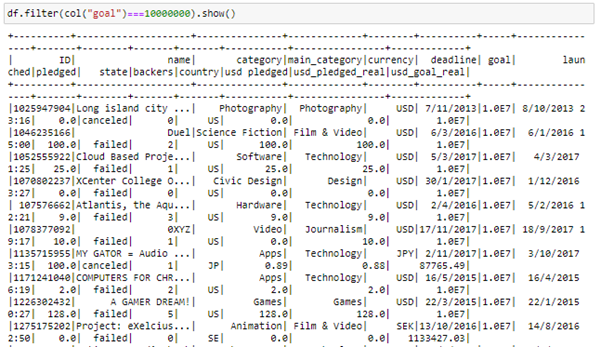


This data entry corresponds to a genuine, successful Kickstarter campaign for a smart watch (<https://www.kickstarter.com/projects/getpebble/pebble-time-awesome-smartwatch-no-compromises>) so the data is valid.

For the ‘goal’ column the max is 100 Billion in local currency, this is high but still reasonable as it might just have been an ambitious project.

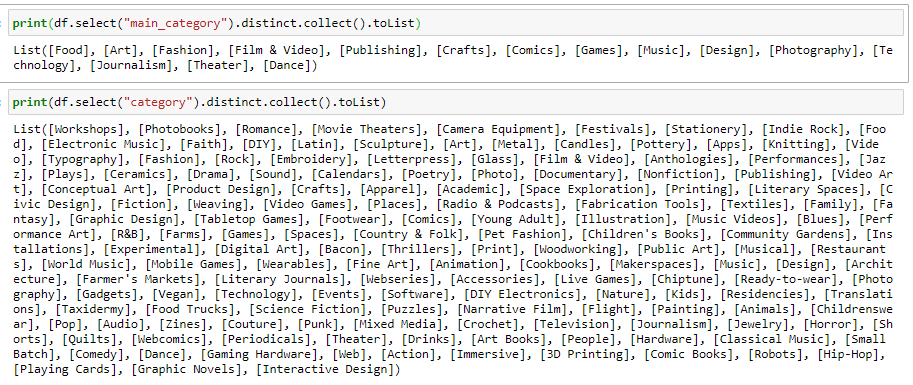
For the ‘backers’ column, we see that there are no nulls and that the min is non negative. The max number is high but on further research this is consistent - the project was the “Exploding Cats” card game with over 219k backers and a ‘successful’ campaign.

The maximum value of the “goal” column is 100000000 or 100M. Investigating the rows containing this value:

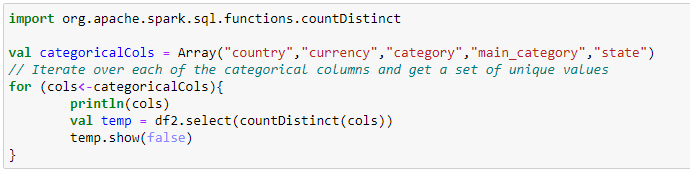
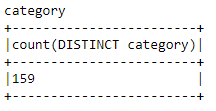


We can see that these rows contain a mix of failed/cancelled projects and some currencies with fractional exchange values against the USD. The failure rate may indicate that the project starters just had very unreasonable goals, but investigating some of the projects reveals them to be genuine.

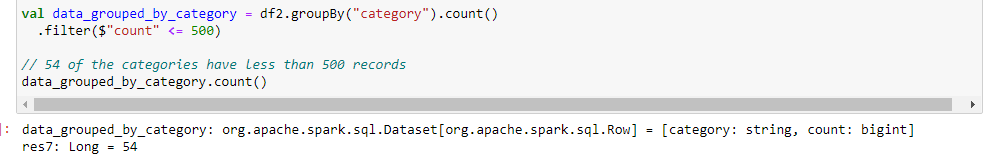
Next we print the unique values for ‘main\_category’ and ‘category’ columns. There aren’t any nulls or invalid values.



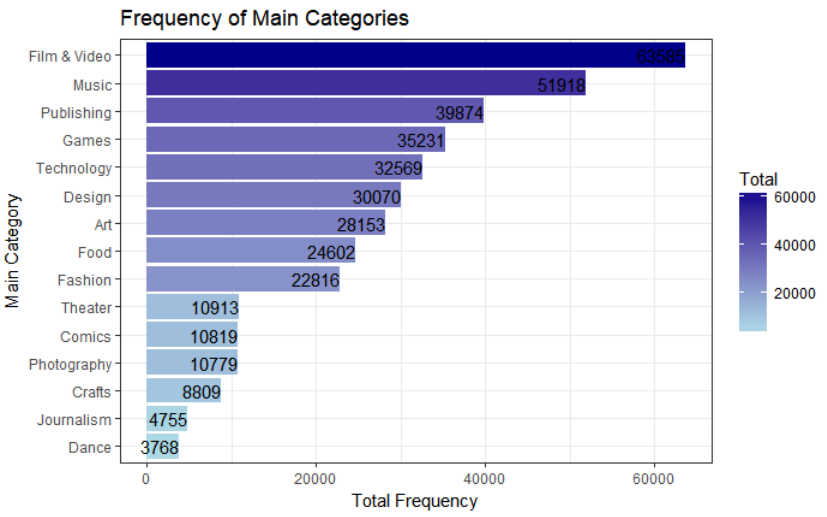
The ‘Category’ categorical variable has 159 values.

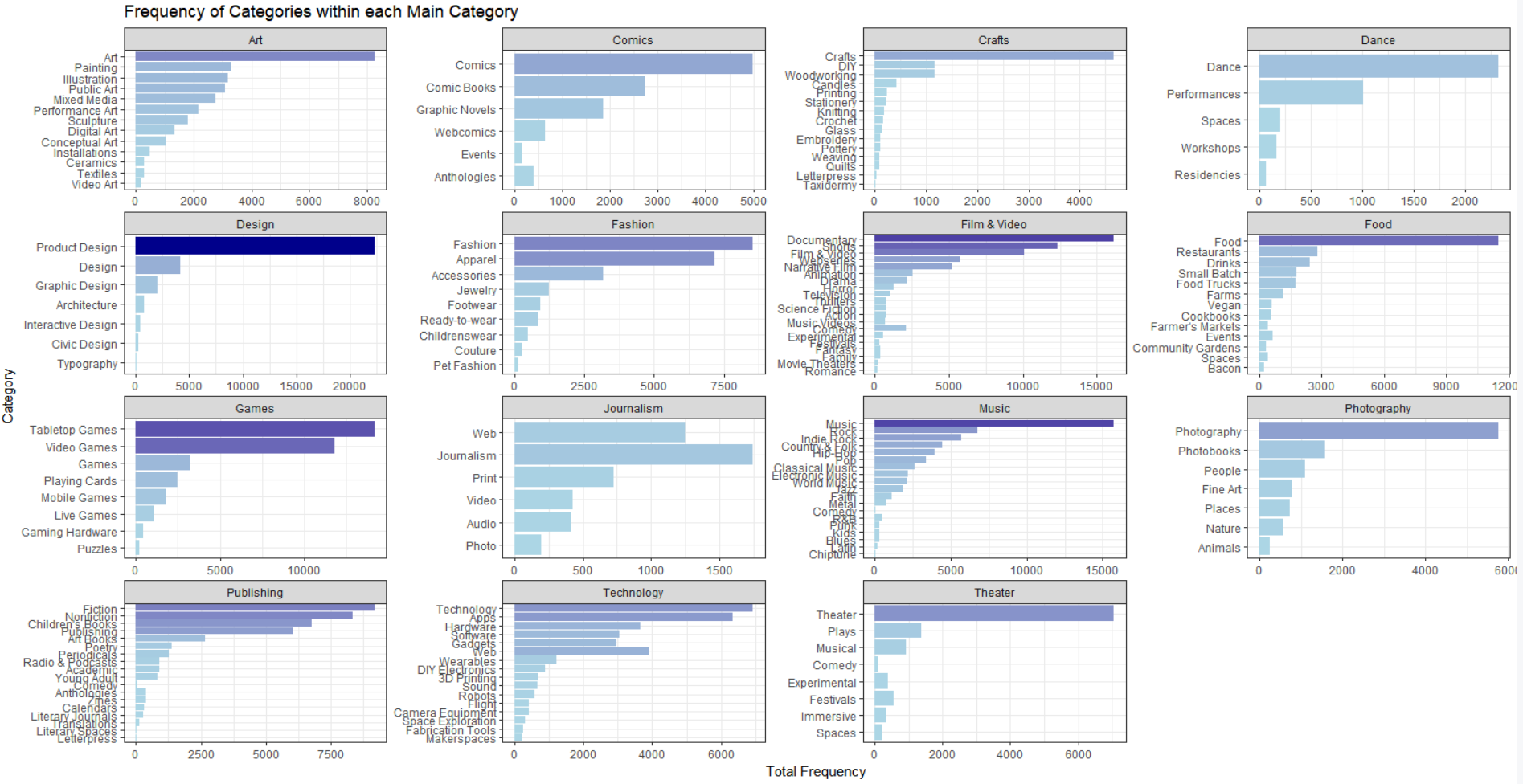
  


54 of the categories have less than 500 records.

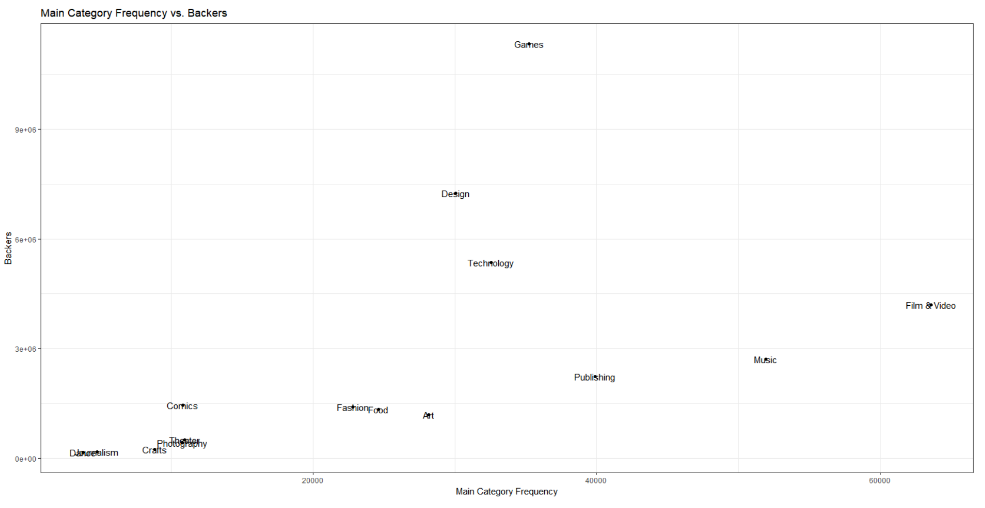


Therefore, this categorical variable may not be as useful as the ‘main\_category’ variable as an input feature for machine learning. The ‘main\_category’ already bins the ‘category’ variable, as seen in this visualisation done through R:

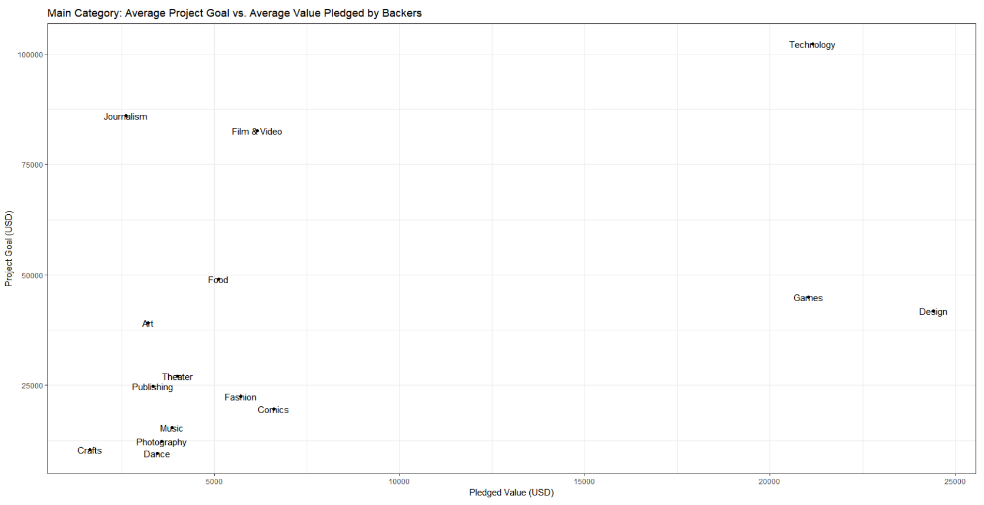




Further visualisations were done to get a feeling of the relationship between categories, backers and money pledged. The below visual, done in R, shows that the Gaming category has the most backers, and is also fourth-mot popular on the side. Other popular campaign categories also tend to get more backers.



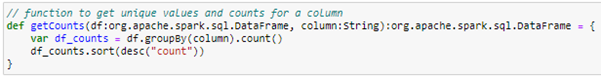
Analysing average goals and average amounts pledged, against category, shows that projects in the Technology space have the highest Goal and second-highest Pledged amount, with Games and Design being the other highest categories for Pledged amounts. This is something to bear in mind for the machine learning feature selection to come.



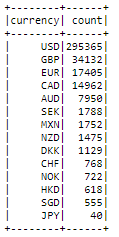
Therefore, we will only transform and dummy the ‘main\_category’ variable later in Step 6.

Next, the dataset has a ‘Currency’ variable which needs to be checked.

A function was defined in order to get all unique values for a column and organise the results by counts in descending order



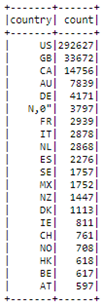
Checking the “currency” column generates the following output:



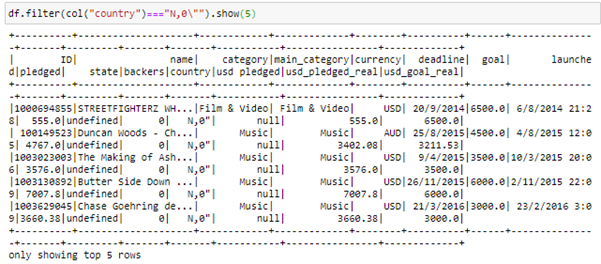
All currencies appear to be valid formats. However, as a failsafe they will need to be compared against a list of valid Kickstarter currencies, and dropped if they don’t match. This is developed in Step 7.

The same analysis is run against the Country column.

Examining the unique values and counts of the country column:

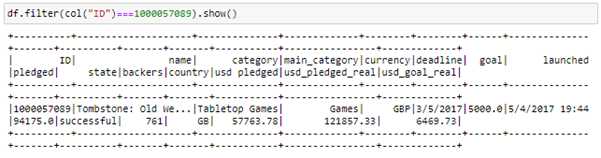


This time we can clearly see that one of the values does not meet the expected format. Inspecting rows that contain ‘N,0”’, we can see that the rows contain undefined and null values, so we will drop these rows. This will be done by also comparing the country list to a list of valid Kickstarter countries, done in Step 7.



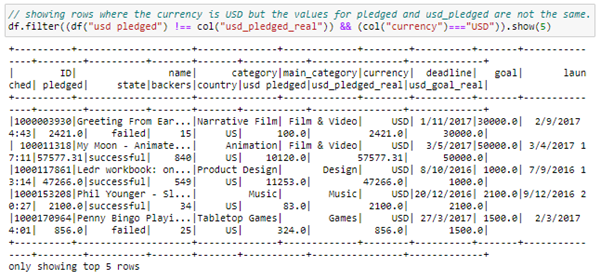
We now look at the ‘usd pledged’ vs ‘usd\_pledged\_real’ columns:

The data source describes ‘usd\_pledged\_real’ and ‘usd\_goal\_real’ as calculated fields based on data from the fixer.io api, and ‘usd pledged’ as the original data from the Kickstarter dataset. Inspecting the data, we can see large discrepancies between ‘usd pledged’ and ‘usd\_pledged\_real’, for example ID 1000057089:



Here we can see that ‘usd pledged’ = 57763.78 and ‘usd\_pledged\_real’ = 121857.33, so which one is accurate/trustworthy?

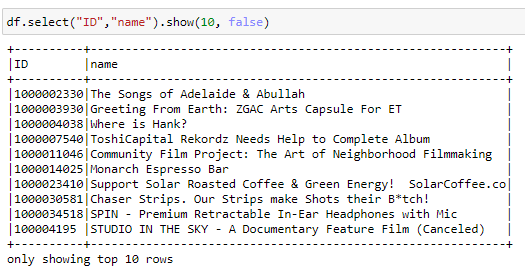
The column ‘usd pledged’ is a USD rate conversion column for projects where the currencies are not set to USD. So projects where USD *is* the project currency should have equal values for pledged and usd\_pledged. By filtering for rows where ‘currency’ = USD and ‘usd\_pledged’ != ‘usd\_pledged\_real’, we should be able to see which one of them is accurate.



We can see that “usd pledged” is wrong in all cases, therefore the ‘usd pledged’ column will be dropped.

# Step 3) Adding feature for project name length

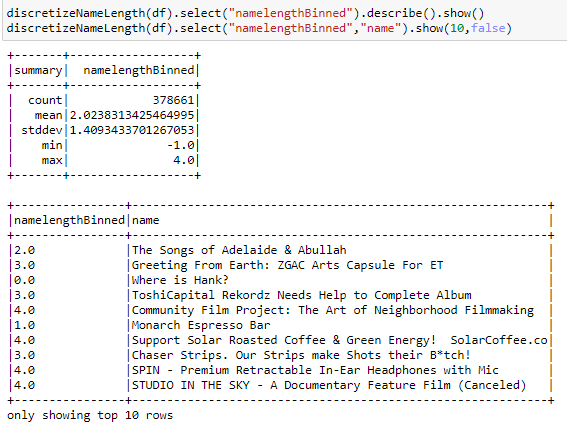
One feature of the dataset that as a group we wanted to explore was the name of the projects. The project name is the face of the project to the world, the first thing that people see before they pledge money to support the project. While the project name is usually quite brief, just a handful of words to catch the eye, some are quite long and detailed. See the first 10 names in the dataset below.



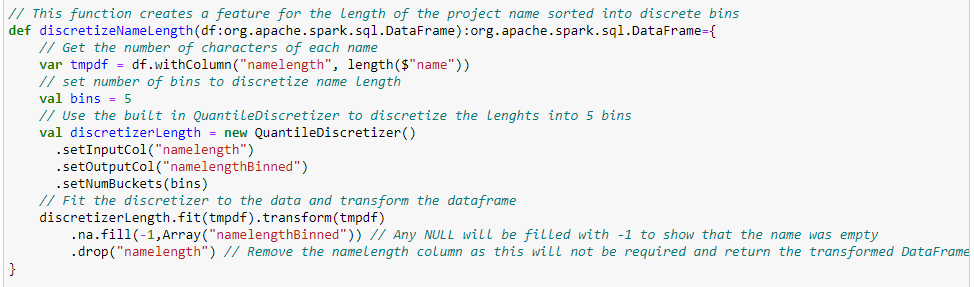
We want to investigate if there is correlation between name length and the success of the project. Does a long-winded name turn off prospective backers? Does a short snappy name attract more support? To do this a new feature has been generated to create name length into a discrete variable from 0 to 4, binned by quantiles.

In the dataset there were only four null values for “name”. These could have been dropped, because we have plenty of data, however it is just as easy to handle this by setting the binned name length to -1 to signal that the name was null.

The ‘nameLengthBinned’ feature as seen below has a min of -1 and a max of 4 as expected, and in the first 10 rows of the dataset the binned name visually length fits in with the names.



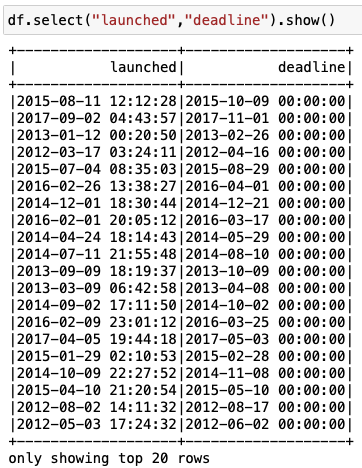
The full function as used in the pipeline:



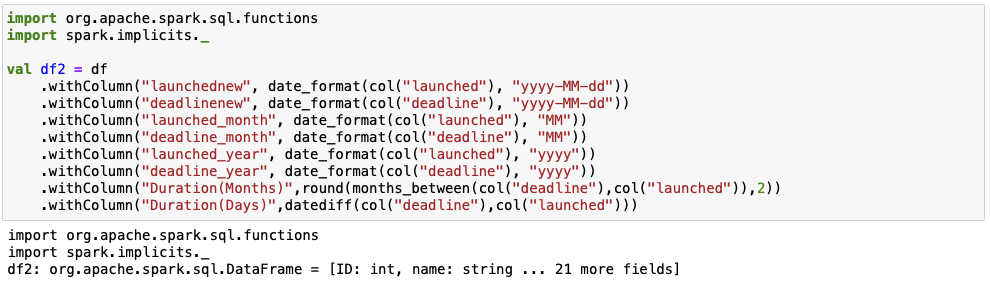
# Step 4) Manipulate dates and time

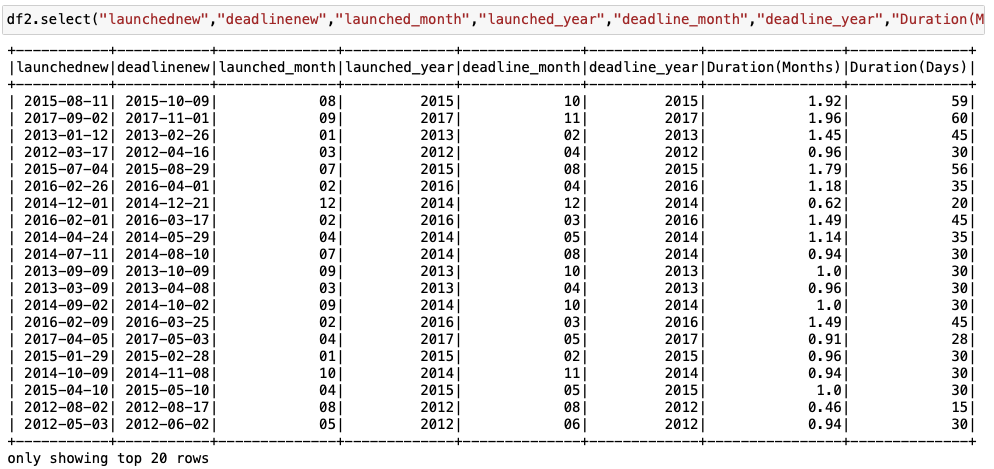
The dataset contained two sets of dates, the launched and deadline date. The project date and time is an important part of the project processes and milestones. The raw dates were recorded as YYYY-MM-DD hh-mm-ss. The datetime format for launched and deadline were not consistent and required standardising.

Project launched and deadline dates:



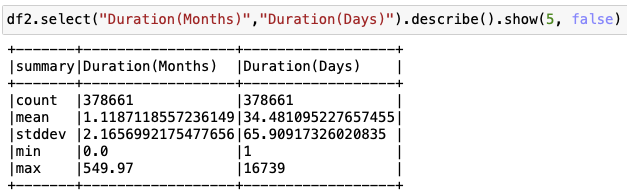
We extract months and years. And from the datetime, we compute durations (‘deadline’ minus ‘launched date’, in days):



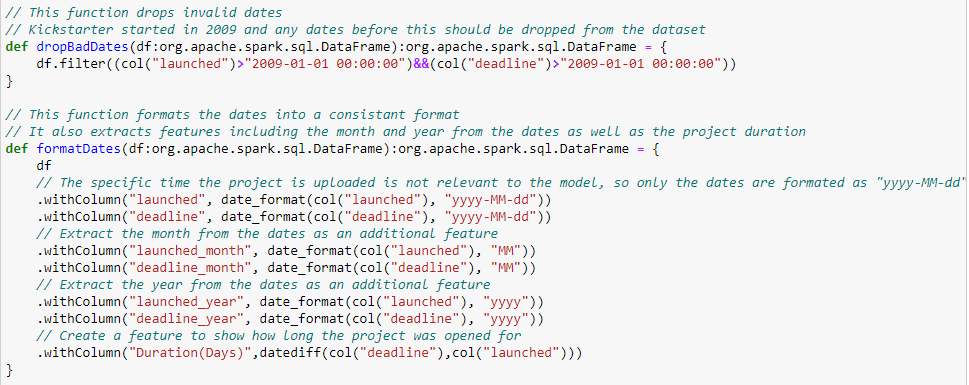


We also check launched dates occur before deadline.

Since the min value for date difference is non-negative we can conclude that all project ‘launch’ dates exceed ‘deadline’ dates.



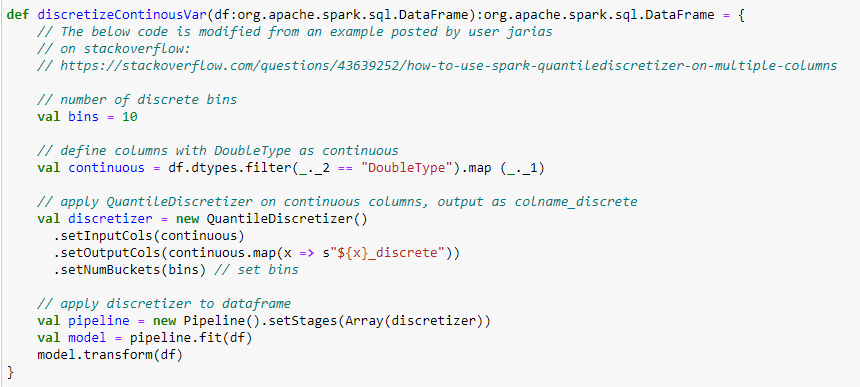
We will encode the date formatting in the pipeline as so, and also make a date filter to drop dates before 2009 (i.e. the date Kickstarter began).



# Step 5) Encode continuous variables

To standardise continuous variables and improve them as a feature in upcoming machine learning models, we will use a function to bin these values into discrete buckets between 0 and 9.

The below code, implemented in the pipeline, checks the dataframe for continuous variables (i.e. those of data type DoubleType). The QuantileDescretizer function bins the values of these columns according to the number of buckets and the changes are made across all continuous columns via the pipeline.



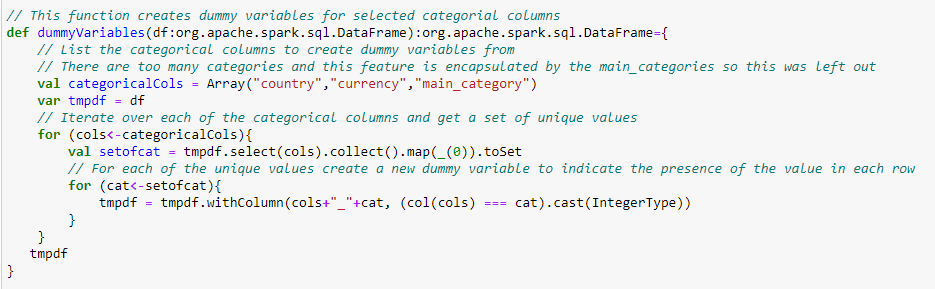
# Step 6) Encode categorical variables

The dataset contains four categorical variables which may be useful for modelling the success or failure of a Kickstarter project, namely:

1. Category;
2. Main category;
3. Currency; and
4. Country.

These variables are all nominal (non-ordered category), and are to be encoded into dummy variables for future modelling. The ‘Category’ variable was not encoded, as it contained too many values, many of which had low counts (see Step 2).

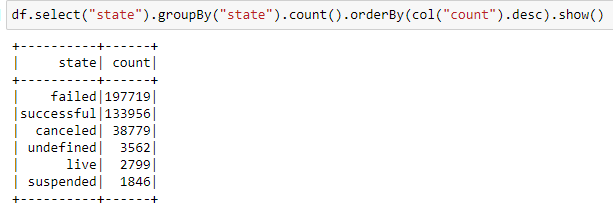
To do the dummy encoding, we created an array called `categoricalCols` which consisted of our dataframe columns of interest. Then we collected each element of the category columns and converted them to a set so that it would return only unique categories. For each of these elements, a binary representation of the original categorical column was created and added to the DataFrame, as shown below.



The output for each categorical column is shown below.

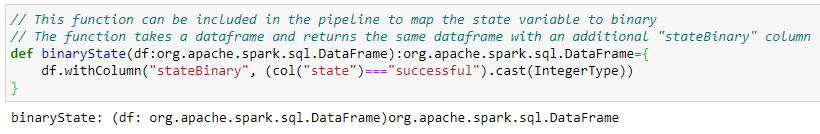
|  |  |
| --- | --- |
| **Column Name** | **Output Post-Encoding** |
| Main category |  |
| Currency |  |
| Country |  |

The dataset also contains a `state` variable, which describes the state of each Kickstarter project, as shown below.

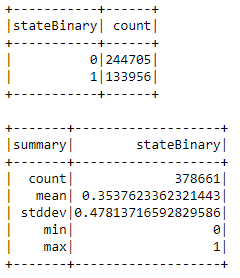


We wish to encode these states into binary data types in a variable called `stateBinary`, where successful states are encoded as 1 and all other states are encoded as 0. Live Kickstarter projects were removed from the dataset, as their success or failure cannot be determined with current data. This is demonstrated by the 2016 vs 2018 comparison in Step 1, where the backers numbers in 2016 data increased in the 2018 dataset for ‘live’ campaigns.

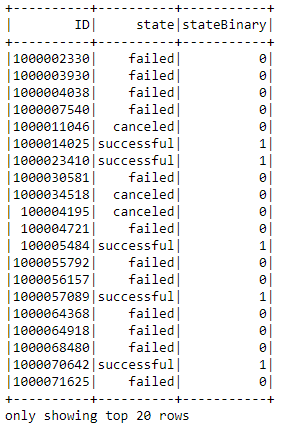
Encoding was performed in the following function, shown with resulting enumerations and descriptive statistics.







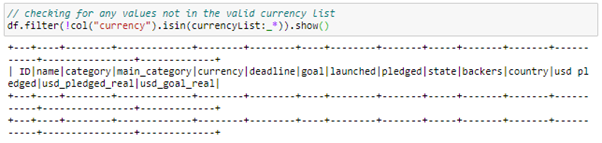
The output of `stateBinary` is shown below.



# Step 7) Drop Invalid Currency and Country Values

A list of valid Kickstarter currencies was generated (according to <https://www.kickstarter.com/blog/new-view-kickstarter-in-your-currency>), to check for invalid currencies in the data.

To check currencies against the valid currency list, the following piece of code returns all rows for which the currency value is not in the valid currency list, as the results show, there are no invalid currencies:



In the case of an invalid currency, the pipeline drops all rows where the currency value is not in the valid currencies list:



Likewise for Country, a valid Kickstarter country list is generated (according to<https://help.kickstarter.com/hc/en-us/articles/115005128594-Who-can-use-Kickstarter->) to check against invalid country entries.





Both checks are amalgamated into a function to check for valid Currency and Country values. It will remove all rows that do not meet the specifications.

