Sections required in your report:

- Main objective of the analysis that specifies whether your model will be focused on prediction or interpretation and the benefits that your analysis provides to the business or stakeholders of this data.
- Brief description of the data set you chose, a summary of its attributes, and an outline of what you are trying to accomplish with this analysis.
- Brief summary of data exploration and actions taken for data cleaning and feature engineering.
- Summary of training at least three different classifier models, preferably of different nature in explainability and predictability. For example, you can start with a simple logistic regression as a baseline, adding other models or ensemble models. Preferably, all your models use the same training and test splits, or the same cross-validation method.
- A paragraph explaining which of your classifier models you recommend as a final model that best fits your needs in terms of accuracy and explainability.
- Summary Key Findings and Insights, which walks your reader through the main drivers of your model and insights from your data derived from your classifier model.
- Suggestions for next steps in analyzing this data, which may include suggesting revisiting this model after adding specific data features that may help you achieve a better explanation or a better prediction

1) Main objective of the analysis that specifies whether your model will be focused on prediction or interpretation and the benefits that your analysis provides to the business or stakeholders of this data.

The goal I was aiming to achieve was to build a model of best possible prediction capabilities.
Since there is a negative relation between interpretation and said prediction, interpretability at the end might have had suffer.
In a regard to above as the best model I have decised to choose the one which exhibits the largest F1-Score on Test data set, together with lowest possible number of predictors used (curse of dimensionality).

Regardless of challenges connected to the interpretability and magnitute of parameters, well designed model should have prove usefull in assessing if your Kickstarter project will turn out to be successful before it even starts.
Among others the likelihood of success is determined by general category of project (music, movie, video game etc.), the size of Goal set as well the number of supporters.

2) Brief description of the data set you chose, a summary of its attributes, and an outline of what you are trying to accomplish with this analysis.

The ks-projects-201801 data set used here in the analysis comes from Kaggle:
[https://www.kaggle.com/kemical/kickstarter-projects (https://www.kaggle.com/kemical/kickstarter-projects)](https://www.kaggle.com/kemical/kickstarter-projects)
Among others ks-projects data set shows attributes of failed, live, and successful Kickstater projects.
By dropping failed projects out of my sample set and by flagging successful projects as Target I was trying to determine which attributes determine successful project. In a result to above I have defined my target as binary. Hence close to 35% of all projects were defined as successful.

Altogether the data set is made of 378 661 rows and 11 columns.
By dropping "live" projects and correlated columns (to avoid multicollinearity) I have been left with 375 862 rows and 6 columns.
Among few categorical columns treated with One-Hot Enoding the data set exhibits extremly skewed numerical variables asking for transformation.

Initial shape of data set:

In [70]:

```
ks_projects.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 375862 entries, (1000002330, 2015-08-11 12:12:28, The Songs of Ad
elaide & Abullah, 2015-10-09) to (999988282, 2011-07-19 09:07:47, Nou Renmen
Ayiti!  We Love Haiti!, 2011-08-16)
Data columns (total 12 columns):
category            375862 non-null object
main_category       375862 non-null object
currency            375862 non-null object
goal                375862 non-null float64
pledged             375862 non-null float64
state               375862 non-null object
backers             375862 non-null float64
country             375862 non-null object
usd pledged         372066 non-null float64
usd_pledged_real    375862 non-null float64
usd_goal_real       375862 non-null float64
successful          375862 non-null int32
dtypes: float64(6), int32(1), object(5)
memory usage: 46.7+ MB
```

More details on [GitHub (https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)](https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)

3) Brief summary of data exploration and actions taken for data cleaning and feature engineering.

From data set we can tell that:

- "Product Design" is the most frequent category when there is 159 unique categories
- "Film & Video" is the most frequent main_category when there is 15 unique main categories
- USD is the most common currency
- 5000 USD is the most common goal
- US is the most frequent country of origin

Since for a design of the data pre-processing steps I used scikit-learn Pipeline() I added following transformers for numeric variables:

- SimpleImputer() with strategy set as "median"
- StandardScaler()
- VarianceThreshold() with threshold set as (.9 * (1 - .9)

In case of categorical variables I added following pre-processing steps:

- SimpleImputer() with strategy set as "most_frequent"
- OneHotEncoder() to get dummy variables
- VarianceThreshold() with threshold set as (.9 * (1 - .9)

The frequencies of unique values per each one of variables:

In [71]:

```
count_unique_values(dataframe=ks_projects
                    ,variables=ks_projects.columns)
```

```
category count distinct:
Product Design     22077
Documentary        16082
Music              15647
Tabletop Games     14072
Shorts             12311
                   ...
Letterpress           48
Chiptune              35
Literary Spaces       23
Taxidermy             13
nan                    0
Length: 160, dtype: int64

main_category count distinct:
Film & Video    63253
Music           51637
Publishing      39575
Games           34944
Technology      32192
Design          29765
Art             27959
Food            24418
Fashion         22566
Theater         10872
Comics          10743
Photography     10731
Crafts           8733
Journalism       4724
Dance            3750
nan                 0
dtype: int64

currency count distinct:
USD    293624
GBP     33853
EUR     17076
CAD     14830
AUD      7880
SEK      1768
MXN      1645
NZD      1464
DKK      1113
CHF       754
NOK       714
HKD       583
SGD       527
JPY        31
nan         0
dtype: int64

goal count distinct:
5000.0     29560
10000.0    25798
1000.0     16838
3000.0     15646
2000.0     15163
```

```
            ...
28962.0        1
46213.0        1
14479.0        1
208.0          1
nan            0
Length: 8303, dtype: int64

pledged count distinct:
0.0         51978
1.0          9157
10.0         4981
25.0         3948
50.0         3608
             ...
39514.0         1
681.13          1
104426.0        1
35452.0         1
nan             0
Length: 61907, dtype: int64

state count distinct:
failed        197719
successful    133956
canceled       38779
undefined       3562
suspended       1846
nan                0
dtype: int64

backers count distinct:
0.0       55060
1.0       34531
2.0       22996
3.0       15929
4.0       11954
           ...
3222.0        1
2745.0        1
5491.0        1
4095.0        1
nan           0
Length: 3953, dtype: int64

country count distinct:
US      290887
GB       33393
CA       14624
AU        7769
DE        4096
N,0"      3796
FR        2887
NL        2833
IT        2802
ES        2224
SE        1737
```

```
MX        1645
NZ        1436
DK        1097
IE         800
CH         747
NO         700
BE         605
HK         583
AT         582
SG         527
LU          61
JP          31
nan          0
dtype: int64

usd pledged count distinct:
0.0         67095
1.0          5316
25.0         3844
10.0         3583
50.0         3117
            ...
10315.88        1
1615.79         1
13251.25        1
25122.0         1
nan          3796
Length: 95037, dtype: int64

usd_pledged_real count distinct:
0.0         51978
1.0          6652
10.0         3596
25.0         3416
50.0         2922
            ...
89186.36        1
26554.0         1
22352.0         1
13173.79        1
nan             0
Length: 105351, dtype: int64

usd_goal_real count distinct:
5000.0      24025
10000.0     20615
1000.0      12960
3000.0      12640
2000.0      11858
            ...
3065.6          1
5016.37         1
56146.13        1
12207.25        1
nan             0
Length: 49885, dtype: int64
```

```
successful count distinct:
0      241906
1      133956
nan         0
dtype: int64
```

More details on [GitHub (https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)](https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)

4) Summary of training at least three different classifier models, preferably of different nature in explainability and predictability. For example, you can start with a simple logistic regression as a baseline, adding other models or ensemble models. Preferably, all your models use the same training and test splits, or the same cross-validation method.

StratifiedShuffleSplit() was used to keep shapes of variables distributions between Train and Test data sets.

Multiple classifying algorithms were used to derive strongest classifier.
To compare models F1-Score have been chosen as performance measure.
Higher the value of F1-Score the better.

However an initial assassment of models was done basing on Accuracy scores:

Linear:

- LogisticRegression Model Score: 0.8749
- RidgeClassifier Model Score: 0.6472
- SGDClassifier Model Score: 0.8138

Trees:

- DecisionTreeClassifier Model Score: 0.9931

Ensamble:

- RandomForestClassifier Model Score: 0.9947
- GradientBoostingClassifier Model Score: 0.9869
- ExtraTreesClassifier Model Score: 0.9913
- BaggingClassifier Model Score: 0.9956

XGBoost:

- XGBClassifier Model Score: 0.9961

F1-Score on Test dat set for the best model basing on Accuracy turned out to be around ~0.95:

In [72]:

```
score_df = pd.DataFrame({'accuracy': accuracy_score(y_test, y_pred),
                         'precision': precision_score(y_test, y_pred),
                         'recall': recall_score(y_test, y_pred),
                         'f1': f1_score(y_test, y_pred),
                         'auc': roc_auc_score(y_test, y_pred)},
                         index=pd.Index([0]))

print(score_df)
```

```
   accuracy  precision  recall    f1   auc
0     0.996      0.990   0.999 0.995 0.997
```

More details on [GitHub (https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)](https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)

5) A paragraph explaining which of your classifier models you recommend as a final model that best fits your needs in terms of accuracy and explainability.

Since major goal of the analysis was prediction I recommend as final model the one with highest F1-Score. That is Extreme Gradient Boosting Classifier.

6) Summary Key Findings and Insights, which walks your reader through the main drivers of your model and insights from your data derived from your classifier model.

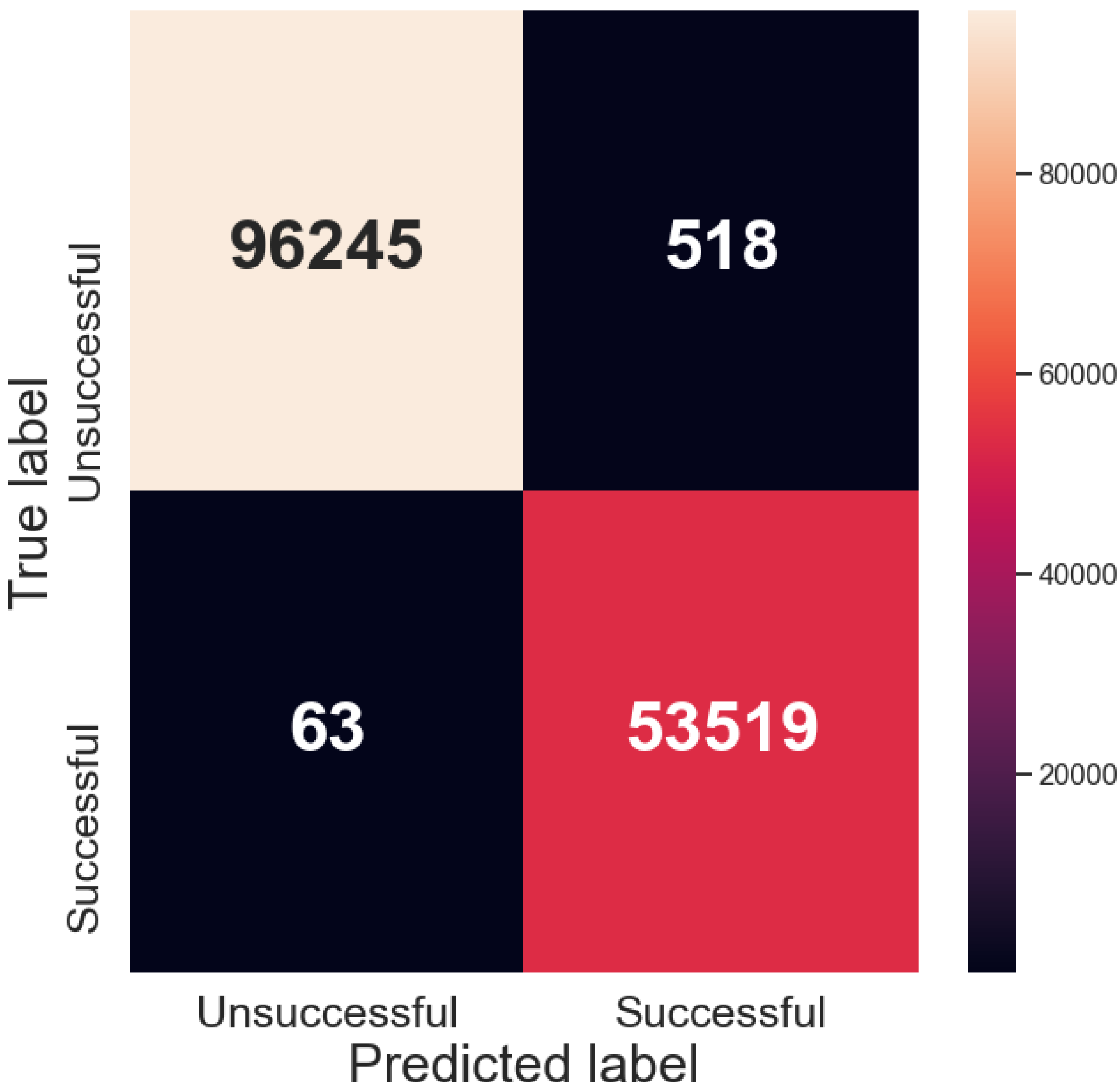
From the Confusion Matrix and F1-Score we can conclude that the model is performing very well. Surprisingly well. Most likely we were lucky to find very strong predictors in our data set.

In [73]:

```
_, ax = plt.subplots(figsize=(10,10))
ax = sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', annot_kws={"size":
40, "weight": "bold"})
labels = ['Unsuccessful', 'Successful']
ax.set_xticklabels(labels, fontsize=25);
ax.set_yticklabels(labels, fontsize=25);
ax.set_ylabel('True label', fontsize=30);
ax.set_xlabel('Predicted label', fontsize=30)
```

Out[73]:

```
Text(0.5, 58.5, 'Predicted label')
```



What surprises even more is once I trained classifiers on just 1000 rows the metrics like F1-Score were still really good:

- accuracy: 0.973
- precision: 0.954
- recall: 0.971
- F1: 0.962
- auc: 0.972

In my opinion such experiment crosses out the assumption about possible overfitting.

Long story short.
If you are a music composer you can most likely try to get funding for your next album via Kickstarter

More details on [GitHub (https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)](https://github.com/KonuTech/kickstarter-projects-classification-techniques/blob/main/classification.ipynb)

7) Suggestions for next steps in analyzing this data, which may include suggesting revisiting this model after adding specific data features that may help you achieve a better explanation or a better prediction.

As next steps on the path of finding the best classification model for finding if Kickstarter project will turn successful I would like to:

- test more Feature Engineering methods
- test more Features Selection methods
- test more hyperparametrs tuning options
- try diffrent algorithms like neural networks/deep learning

However the models built are already very strong. The improvement on correct classifiaction probably would not improve much.