Configuration Manual

MSc Research Project

Master of Science in FinTech

Kanishka Dhyani

Student ID: X22232745

School of Computing

National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland

MSc Project Submission Sheet

Student Name: Kanishka Dhyani

Student ID: X22232745

Programme: Master of Science in FinTech **Year:** 2024

Module: MSc. Research project

Lecture: Victor Del Rosal

Submission Due Date: 12th August, 2024

Project Title: Unveiling The Key Attributes of Leading Crowdfunding Projects

Word Count: 662 **Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Kanishka Dhyani

Date: 12th Aug 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties**.
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Kanishka Dhyani

X22232745

1. General Outlook

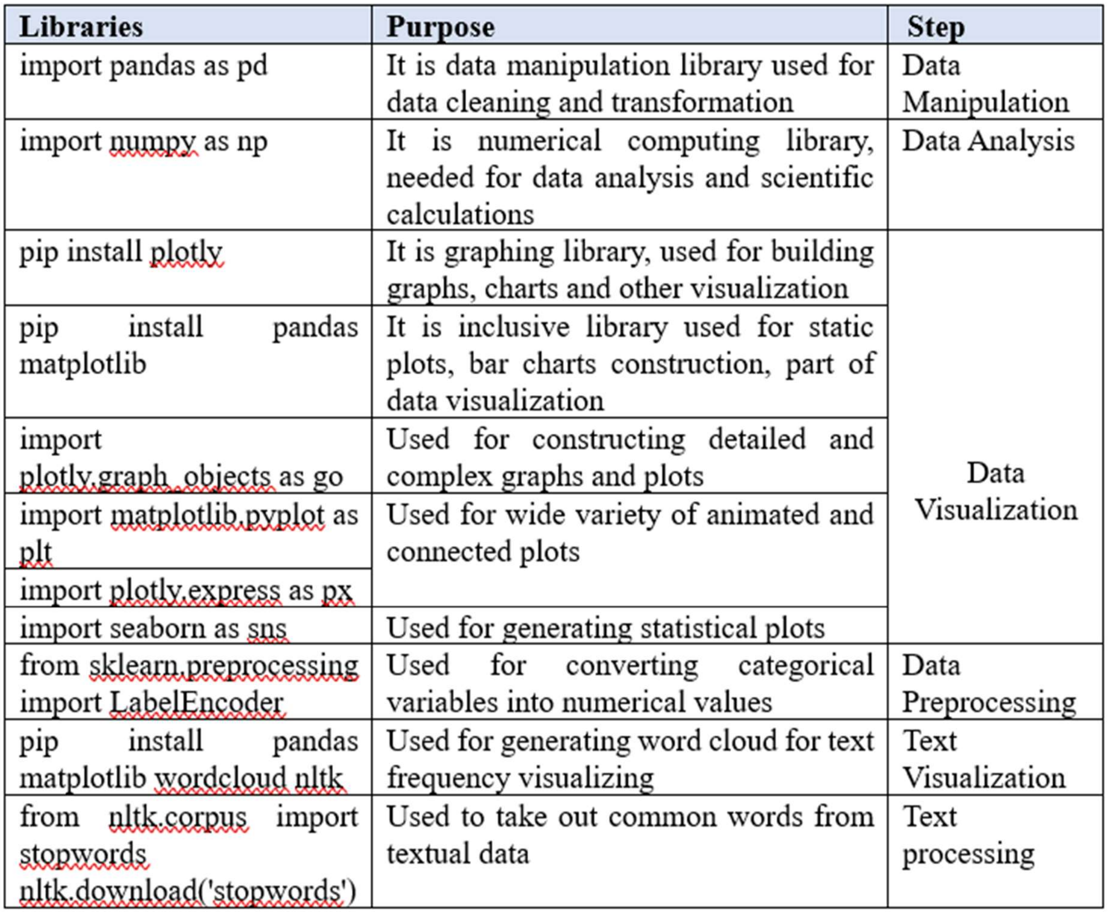
In the study, implementation of Machine Learning models and other relevant visualization are conducted on Google Collab using Python. This manual furnishes a report including technical set ups of software and hardware required to complete this research under the title of: ‘Unveiling The Key Attributes of Leading Crowdfunding Projects’.

2. Data and Libraries

The data was collected from ‘Web Robots’ having scraper robot this website collects monthly data from the website of Kickstarter. All the 110 files from 2009 to 2024(till 15.06.2024) were first downloaded and then collated into single Masterfile

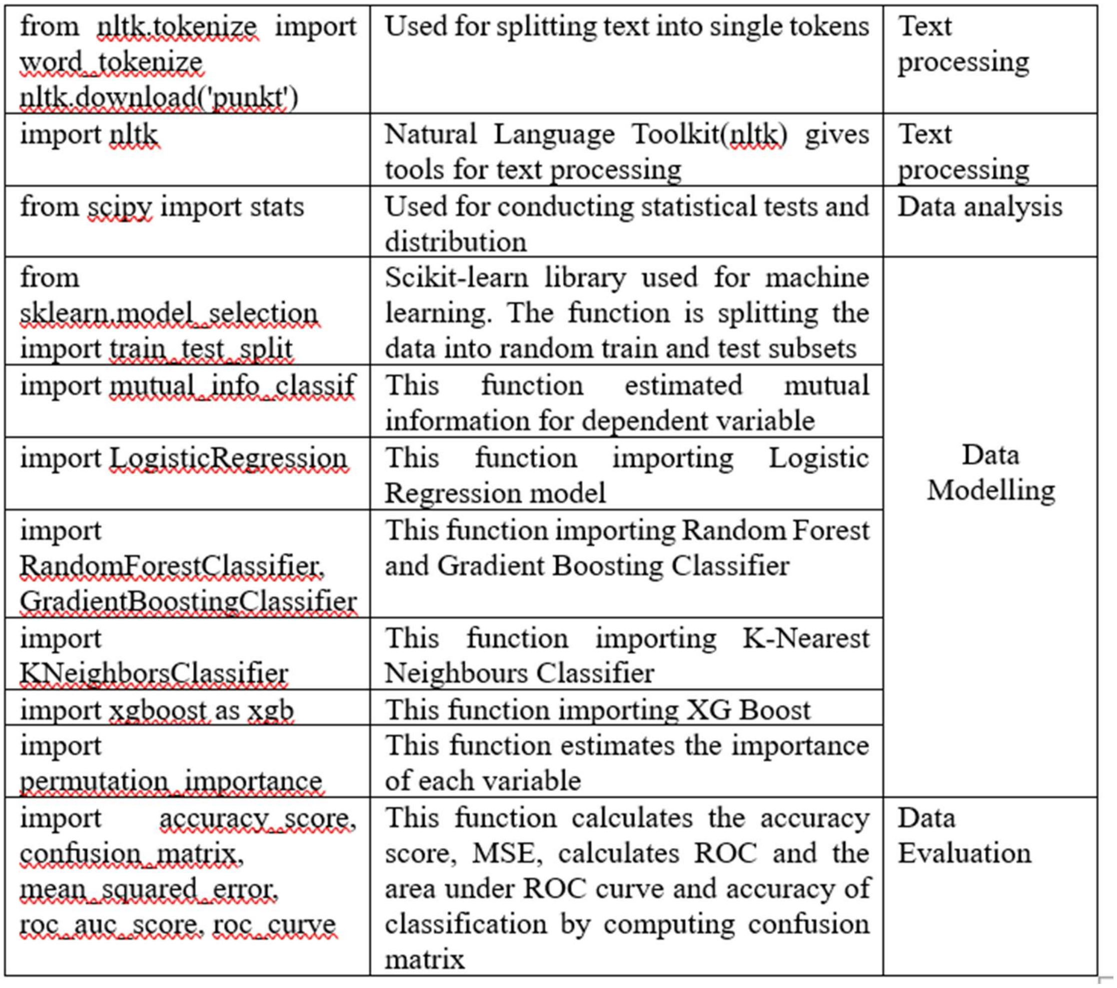
- The website link for this site: https://webrobots.io/kickstarter-datasets/

Extending the capabilities of programming language, we have used below mentioned Python libraries (pre-written codes collection):



Libraries	Purpose	Step
import pandas as pd	It is data manipulation library used for data cleaning and transformation	Data Manipulation
import numpy as np	It is numerical computing library, needed for data analysis and scientific calculations	Data Analysis
pip install plotly	It is graphing library, used for building graphs, charts and other visualization	Data Visualization
pip install pandas matplotlib	It is inclusive library used for static plots, bar charts construction, part of data visualization	
import plotly.graph_objects as go	Used for constructing detailed and complex graphs and plots	
import matplotlib.pyplot as plt	Used for wide variety of animated and connected plots	
import plotly.express as px		
import seaborn as sns	Used for generating statistical plots	
from sklearn.preprocessing import LabelEncoder	Used for converting categorical variables into numerical values	Data Preprocessing
pip install pandas matplotlib wordcloud nltk	Used for generating word cloud for text frequency visualizing	Text Visualization

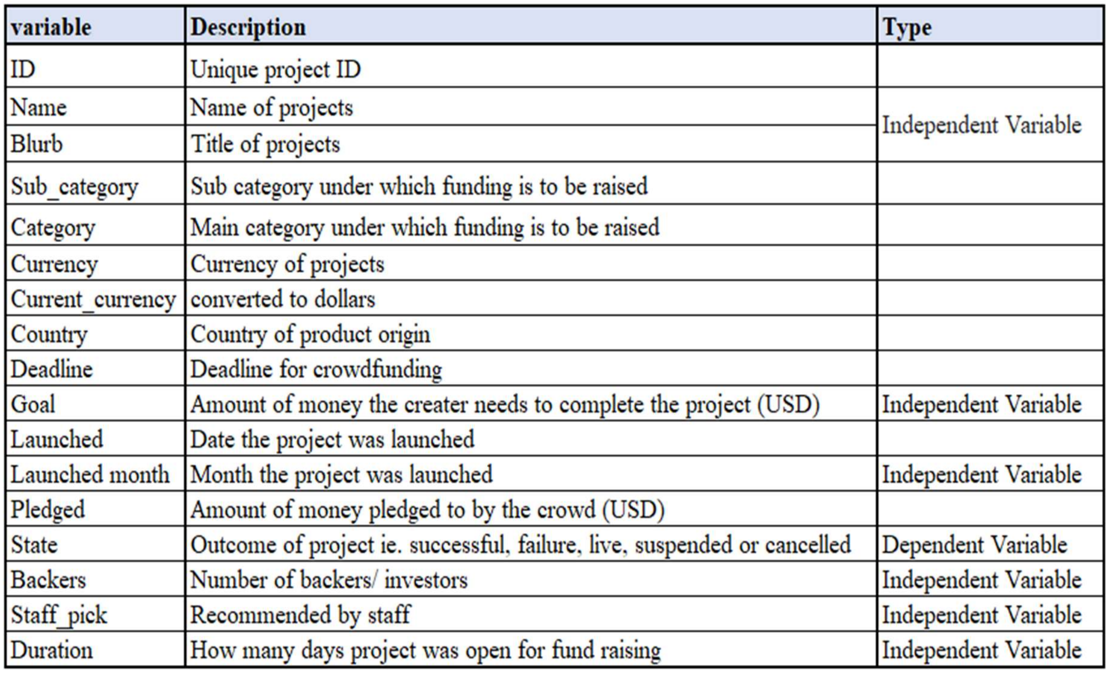
from nltk.corpus import stopwords
nltk.download('stopwords')
	Used to take out common words from textual data	Text processing



from nltk.tokenize import word_tokenize
nltk.download('punkt')
	Used for splitting text into single tokens	Text processing

import nltk	Natural Language Toolkit(nltk) gives tools for text processing	Text processing
from scipy import stats	Used for conducting statistical tests and distribution	Data analysis
from sklearn.model_selection import train_test_split	Scikit-learn library used for machine learning. The function is splitting the data into random train and test subsets	Data Modelling
import mutual_info_classif	This function estimated mutual information for dependent variable	
import LogisticRegression	This function importing Logistic Regression model	
import RandomForestClassifier, GradientBoostingClassifier	This function importing Random Forest and Gradient Boosting Classifier	
import KNeighborsClassifier	This function importing K-Nearest Neighbours Classifier	
import xgboost as xgb	This function importing XG Boost	
import permutation_importance	This function estimates the importance of each variable	
import accuracy_score, confusion_matrix, mean_squared_error, roc_auc_score, roc_curve	This function calculates the accuracy score, MSE, calculates ROC and the area under ROC curve and accuracy of classification by computing confusion matrix	Data Evaluation

- **Metadata**



variable	Description	Type
ID	Unique project ID	
Name	Name of projects	Independent Variable
Blurb	Title of projects	
Sub_category	Sub category under which funding is to be raised	
Category	Main category under which funding is to be raised	
Currency	Currency of projects	
Current_currency	converted to dollars	
Country	Country of product origin	
Deadline	Deadline for crowdfunding	
Goal	Amount of money the creater needs to complete the project (USD)	Independent Variable
Launched	Date the project was launched	
Launched month	Month the project was launched	Independent Variable
Pledged	Amount of money pledged to by the crowd (USD)	
State	Outcome of project ie. successful, failure, live, suspended or cancelled	Dependent Variable
Backers	Number of backers/ investors	Independent Variable
Staff_pick	Recommended by staff	Independent Variable
Duration	How many days project was open for fund raising	Independent Variable

3. System Specification

Hardware Requirement

- Lenovo IdeaPad 3 15IAU7 Laptop- Model
- 12th Gen Intel(R) Core (TM) i5-1235U 1.30 GHz- Processor
- 16.0 GB (15.7 GB usable)- RAM
- 64-bit operating system, x64-based processor- System type
- Windows 11 Home Single Language- operating system

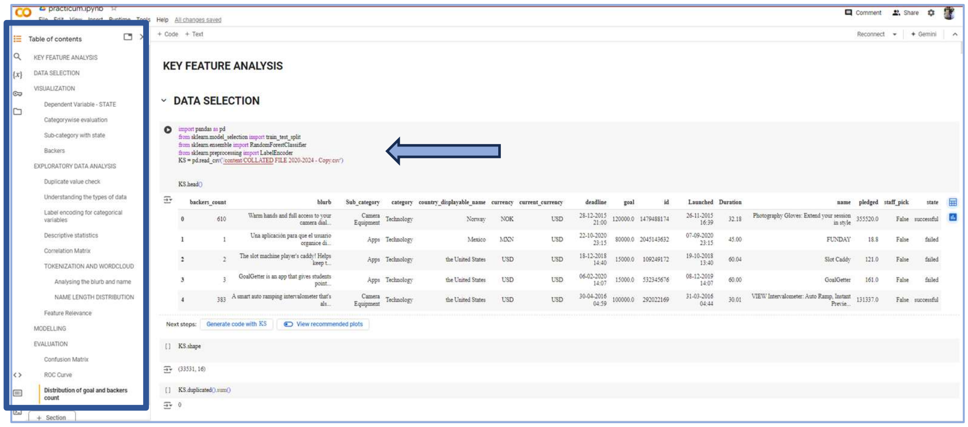
Software Requirement

- Python Programming language and Google Collab

4. Operational Process

The operational process is categorized into several parts: importing data, preprocessing and transformation of data, modelling and evaluation with data analysis. Used codes and applied models are specified as follows: -

The collab notebook is well labelled as seen in the table of content in the left side highlighted and each step has specific heading. Firstly, we imported the data and check the **imported** data's first five datapoints and the shape (no. of rows and columns)



KEY FEATURE ANALYSIS

DATA SELECTION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
KS = pd.read_csv('/content/COLLATED FILE 2020-2024 - Copy.csv')


KS.head()
```

	backers_count	blurb	Sub_category	category	country_displayable_name	currency	current_currency	deadline	goal	id	Launched	Duration	name	pledged	staff_pick	state
0	610	Warm hands and full access to your camera dial...	Camera Equipment	Technology	Norway	NOK	USD	28-12-2015 21:00	120000.0	1479488174	26-11-2015 16:39	32.18	Photography Gloves: Extend your session in style	355520.0	False	successful
1	1	Una aplicación para que el usuario organice di...	Apps	Technology	Mexico	MXN	USD	22-10-2020 23:15	80000.0	2045143632	07-09-2020 23:15	45.00	FUNDAY	18.8	False	failed
2	2	The slot machine player's caddy! Helps keep t...	Apps	Technology	the United States	USD	USD	18-12-2018 14:40	15000.0	109249172	19-10-2018 13:40	60.04	Slot Caddy	121.0	False	failed
3	3	GoalGetter is an app that gives students point...	Apps	Technology	the United States	USD	USD	06-02-2020 14:07	15000.0	532345676	08-12-2019 14:07	60.00	GoalGetter	161.0	False	failed
4	383	A smart auto ramping intervalometer that's als...	Camera Equipment	Technology	the United States	USD	USD	30-04-2016 04:59	100000.0	292022169	31-03-2016 04:44	30.01	VIEW Intervalometer: Auto Ramp, Instant Previe...	131337.0	False	successful

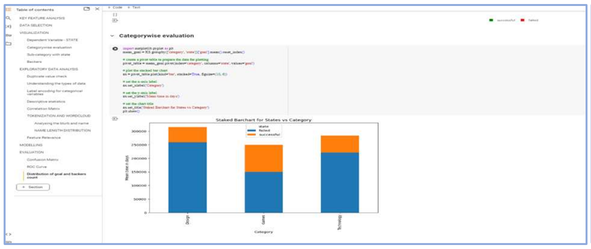
```
KS.shape
```

```
(33531, 16)
```

```
KS.duplicated().sum()
```

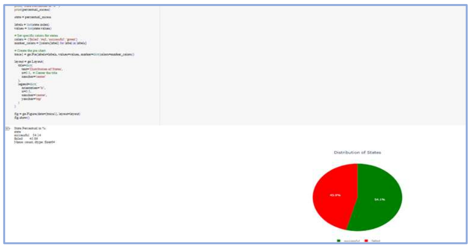
```
0
```

Under **Exploratory Data Analysis,** for **Visualization**, we have plotted graphs and pie charts to understand some relationship between independent and dependent variables:



Categorywise evaluation

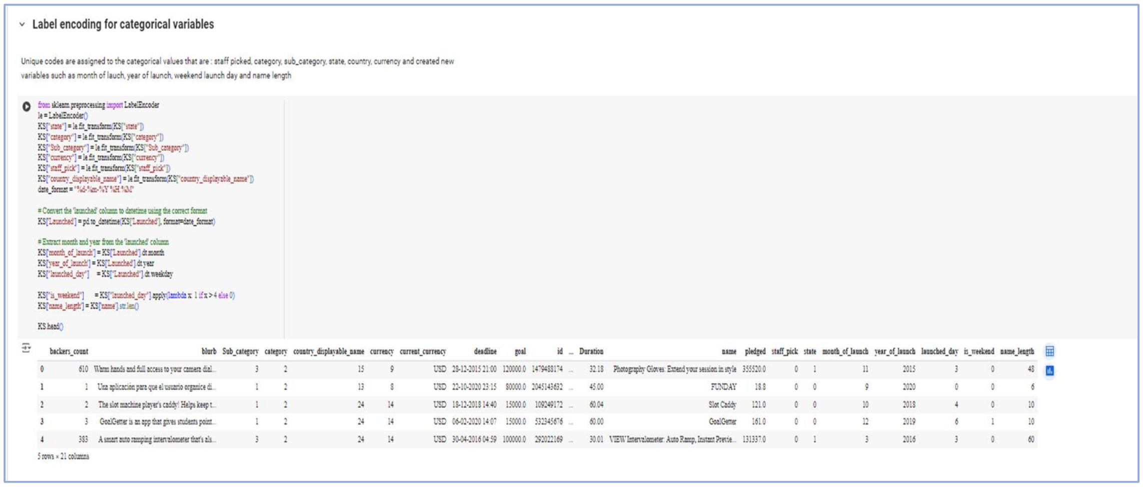
```
import matplotlib.pyplot as plt
mean_goal = KS.groupby(['category', 'state'])['goal'].mean().reset_index()

# create a pivot table to prepare the data for plotting
pivot_table = mean_goal.pivot(index='category', columns='state', values='goal')

# plot the stacked bar chart
ax = pivot_table.plot(kind='bar', stacked=True, figsize=(10, 6))

# set the x-axis label
ax.set_xlabel('Category')

# set the y-axis label
ax.set_ylabel('Mean time in days')

# set the chart title
ax.set_title('Staked Barchart for States vs Category')
plt.show()
```



```
print(percentual_success)

state = percentual_success

labels = list(state.index)
values = list(state.values)

# Set specific colors for states
colors = {'failed': 'red', 'successful': 'green'}
marker_colors = [colors[label] for label in labels]

# Create the pie chart
trace1 = go.Pie(labels=labels, values=values, marker=dict(colors=marker_colors))

layout = go.Layout(
    title=dict(
        text='Distribution of States',
        x=0.5, # Center the title
        xanchor='center'
    ),
    legend=dict(
        orientation='h',
        x=0.5,
        xanchor='center',
        yanchor='top'
    )
)

fig = go.Figure(data=[trace1], layout=layout)
fig.show()
```

```
State Percentual in %:
state
successful    54.14
failed        45.86
Name: count, dtype: float64
```

Checked for duplicates and data types of variables after that label encoding was performed to convert categorical variable to numeric values. Converted variables are: state, category, sub_category, currency, staff_pick, country_displayable_name and created new variable month_of_launch, year_of_launch, launched_day and is_weekend.



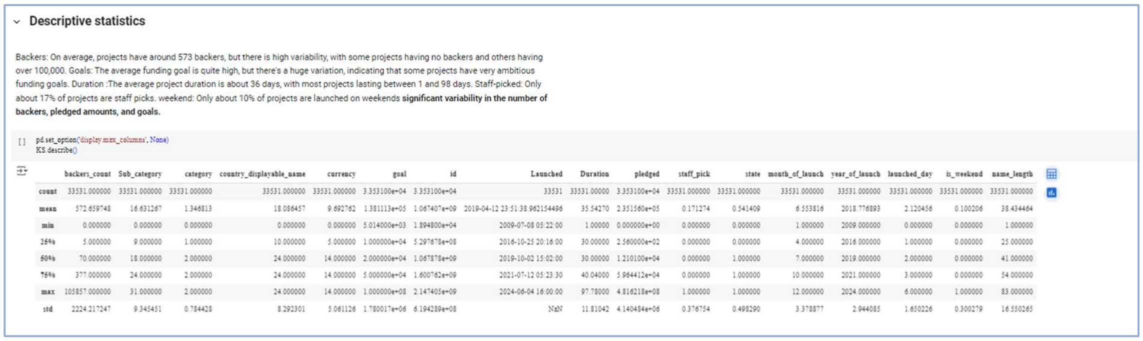
Label encoding for categorical variables

Unique codes are assigned to the categorical values that are : staff picked, category, sub_category, state, country, currency and created new variables such as month of lauch, year of launch, weekend launch day and name length

```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
KS["state"] = le.fit_transform(KS["state"])
KS["category"] = le.fit_transform(KS["category"])
KS["Sub_category"] = le.fit_transform(KS["Sub_category"])
KS["currency"] = le.fit_transform(KS["currency"])
KS["staff_pick"] = le.fit_transform(KS["staff_pick"])
KS["country_displayable_name"] = le.fit_transform(KS["country_displayable_name"])
date_format = "%d-%m-%Y %H:%M"

# Convert the 'launched' column to datetime using the correct format
KS['Launched'] = pd.to_datetime(KS['Launched'], format=date_format)

# Extract month and year from the 'launched' column
KS['month_of_launch'] = KS['Launched'].dt.month
KS['year_of_launch'] = KS['Launched'].dt.year
KS["launched_day"]    = KS["Launched"].dt.weekday

KS["is_weekend"]      = KS["launched_day"].apply(lambda x: 1 if x > 4 else 0)
KS['name_length'] = KS['name'].str.len()

KS.head()
```

	backers_count	blurb	Sub_category	category	country_displayable_name	currency	current_currency	deadline	goal	id	...	Duration	name	pledged	staff_pick	state	month_of_launch	year_of_launch	launched_day	is_weekend	name_length
0	610	Warm hands and full access to your camera dial...	3	2	15	9	USD	28-12-2015 21:00	120000.0	1479488174	...	32.18	Photography Gloves: Extend your session in style	355520.0	0	1	11	2015	3	0	48
1	1	Una aplicación para que el usuario organice di...	1	2	13	8	USD	22-10-2020 23:15	80000.0	2045143632	...	45.00	FUNDAY	18.8	0	0	9	2020	0	0	6
2	2	The slot machine player's caddy! Helps keep t...	1	2	24	14	USD	18-12-2018 14:40	15000.0	109249172	...	60.04	Slot Caddy	121.0	0	0	10	2018	4	0	10
3	3	GoalGetter is an app that gives students point...	1	2	24	14	USD	06-02-2020 14:07	15000.0	532345676	...	60.00	GoalGetter	161.0	0	0	12	2019	6	1	10
4	383	A smart auto ramping intervalometer that's als...	3	2	24	14	USD	30-04-2016 04:59	100000.0	292022169	...	30.01	VIEW Intervalometer: Auto Ramp, Instant Previe...	131337.0	0	1	3	2016	3	0	60

5 rows × 21 columns

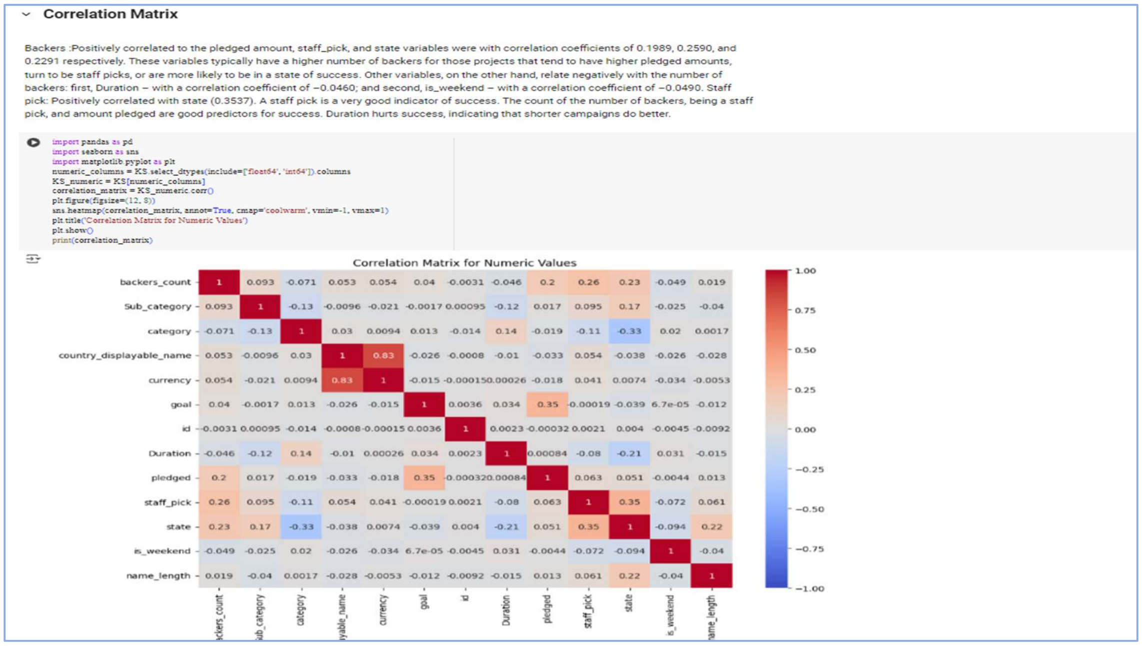


Descriptive statistics

Backers: On average, projects have around 573 backers, but there is high variability, with some projects having no backers and others having over 100,000. Goals: The average funding goal is quite high, but there's a huge variation, indicating that some projects have very ambitious funding goals. Duration :The average project duration is about 36 days, with most projects lasting between 1 and 98 days. Staff-picked: Only about 17% of projects are staff picks. weekend: Only about 10% of projects are launched on weekends **significant variability in the number of backers, pledged amounts, and goals.**

```python
pd.set_option('display.max_columns', None)
KS.describe()
```

	backers_count	Sub_category	category	country_displayable_name	currency	goal	id	Launched	Duration	pledged	staff_pick	state	month_of_launch	year_of_launch	launched_day	is_weekend	name_length
count	33531.000000	33531.000000	33531.000000	33531.000000	33531.000000	3.353100e+04	3.353100e+04	33531	33531.00000	3.353100e+04	33531.000000	33531.000000	33531.000000	33531.000000	33531.000000	33531.000000	33531.000000
mean	572.659748	16.631267	1.346813	18.086457	9.692762	1.381113e+05	1.067407e+09	2019-04-12 23:51:38.962154496	35.54270	2.351560e+05	0.171274	0.541409	6.553816	2018.776893	2.120456	0.100206	38.434464
min	0.000000	0.000000	0.000000	0.000000	0.000000	5.014000e+03	1.894800e+04	2009-07-08 05:22:00	1.00000	0.000000e+00	0.000000	0.000000	1.000000	2009.000000	0.000000	0.000000	1.000000
25%	5.000000	9.000000	1.000000	10.000000	5.000000	1.000000e+04	5.297678e+08	2016-10-25 20:16:00	30.00000	2.560000e+02	0.000000	0.000000	4.000000	2016.000000	1.000000	0.000000	25.000000
50%	70.000000	18.000000	2.000000	24.000000	14.000000	2.000000e+04	1.067878e+09	2019-10-02 15:02:00	30.00000	1.210100e+04	0.000000	1.000000	7.000000	2019.000000	2.000000	0.000000	41.000000
75%	377.000000	24.000000	2.000000	24.000000	14.000000	5.000000e+04	1.600762e+09	2021-07-12 05:23:30	40.04000	5.964412e+04	0.000000	1.000000	10.000000	2021.000000	3.000000	0.000000	54.000000
max	105857.000000	31.000000	2.000000	24.000000	14.000000	1.000000e+08	2.147405e+09	2024-06-04 16:00:00	97.78000	4.816218e+08	1.000000	1.000000	12.000000	2024.000000	6.000000	1.000000	83.000000
std	2224.217247	9.345451	0.784428	8.292301	5.061126	1.780017e+06	6.194289e+08	NaN	11.81042	4.140484e+06	0.376754	0.498290	3.378877	2.944085	1.650226	0.300279	16.550265



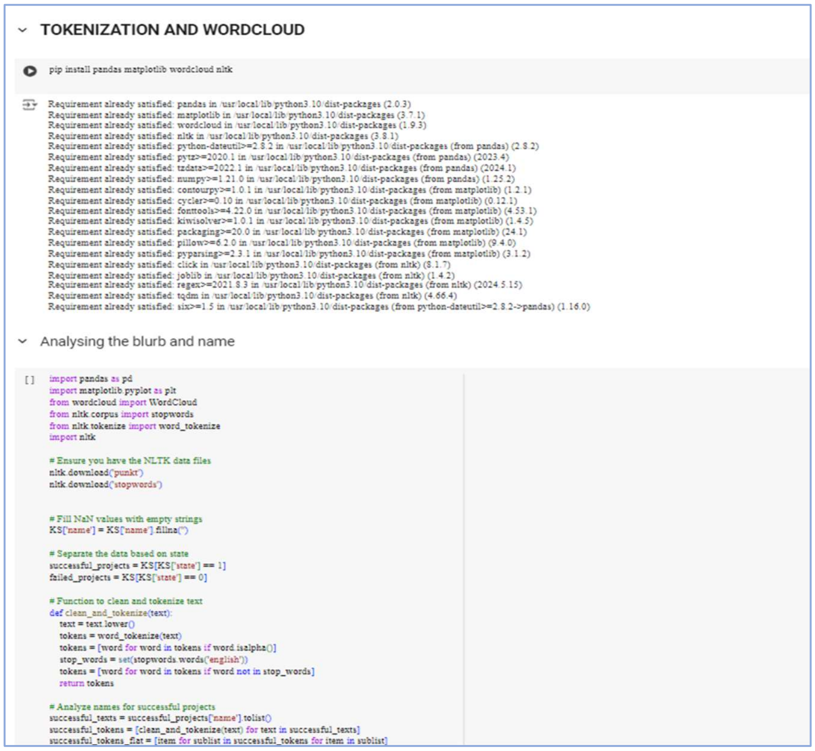
Correlation Matrix

Backers :Positively correlated to the pledged amount, staff_pick, and state variables were with correlation coefficients of 0.1989, 0.2590, and 0.2291 respectively. These variables typically have a higher number of backers for those projects that tend to have higher pledged amounts, turn to be staff picks, or are more likely to be in a state of success. Other variables, on the other hand, relate negatively with the number of backers: first, Duration – with a correlation coefficient of −0.0460; and second, is_weekend – with a correlation coefficient of −0.0490. Staff pick: Positively correlated with state (0.3537). A staff pick is a very good indicator of success. The count of the number of backers, being a staff pick, and amount pledged are good predictors for success. Duration hurts success, indicating that shorter campaigns do better.

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
numeric_columns = KS.select_dtypes(include=['float64', 'int64']).columns
KS_numeric = KS[numeric_columns]
correlation_matrix = KS_numeric.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix for Numeric Values')
plt.show()
print(correlation_matrix)
```

Correlation Matrix for Numeric Values
backers_count 1 0.093 -0.071 0.053 0.054 0.04 -0.0031 -0.046 0.2 0.26 0.23 -0.049 0.019
Sub_category 0.093 1 -0.13 -0.0096 -0.021 -0.0017 0.00095 -0.12 0.017 0.095 0.17 -0.025 -0.04
category -0.071 -0.13 1 0.03 0.0094 0.013 -0.014 0.14 -0.019 -0.11 -0.33 0.02 0.0017
country_displayable_name 0.053 -0.0096 0.03 1 0.83 -0.026 -0.0008 -0.01 -0.033 0.054 -0.038 -0.026 -0.028
currency 0.054 -0.021 0.0094 0.83 1 -0.015 -0.00015 0.00026 -0.018 0.041 0.0074 -0.034 -0.0053
goal 0.04 -0.0017 0.013 -0.026 -0.015 1 0.0036 0.034 0.35 -0.00019 -0.039 6.7e-05 -0.012
id -0.0031 0.00095 -0.014 -0.0008 -0.00015 0.0036 1 0.0023 -0.00032 0.0021 0.004 -0.0045 -0.0092
Duration -0.046 -0.12 0.14 -0.01 0.00026 0.034 0.0023 1 0.00084 -0.08 -0.21 0.031 -0.015
pledged 0.2 0.017 -0.019 -0.033 -0.018 0.35 -0.00032 0.00084 1 0.063 0.051 -0.0044 0.013
staff_pick 0.26 0.095 -0.11 0.054 0.041 -0.00019 0.0021 -0.08 0.063 1 0.35 -0.072 0.061
state 0.23 0.17 -0.33 -0.038 0.0074 -0.039 0.004 -0.21 0.051 0.35 1 -0.094 0.22
is_weekend -0.049 -0.025 0.02 -0.026 -0.034 6.7e-05 -0.0045 0.031 -0.0044 -0.072 -0.094 1 -0.04
name_length 0.019 -0.04 0.0017 -0.028 -0.0053 -0.012 -0.0092 -0.015 0.013 0.061 0.22 -0.04 1
ackers_count
ub_category
category
yable_name
currency
goal
id
Duration
pledged
staff_pick
state
is_weekend
name_length
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Text Analysis



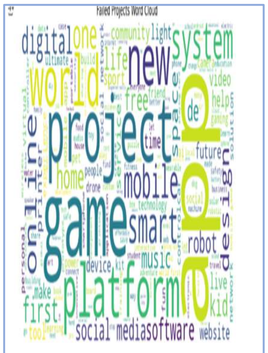
TOKENIZATION AND WORDCLOUD

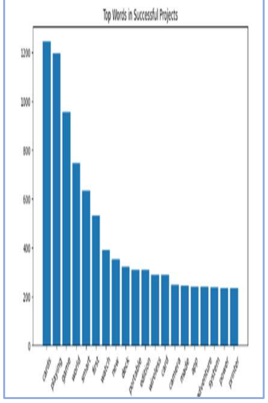
```
pip install pandas matplotlib wordcloud nltk
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-packages (1.9.3)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.25.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.5.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

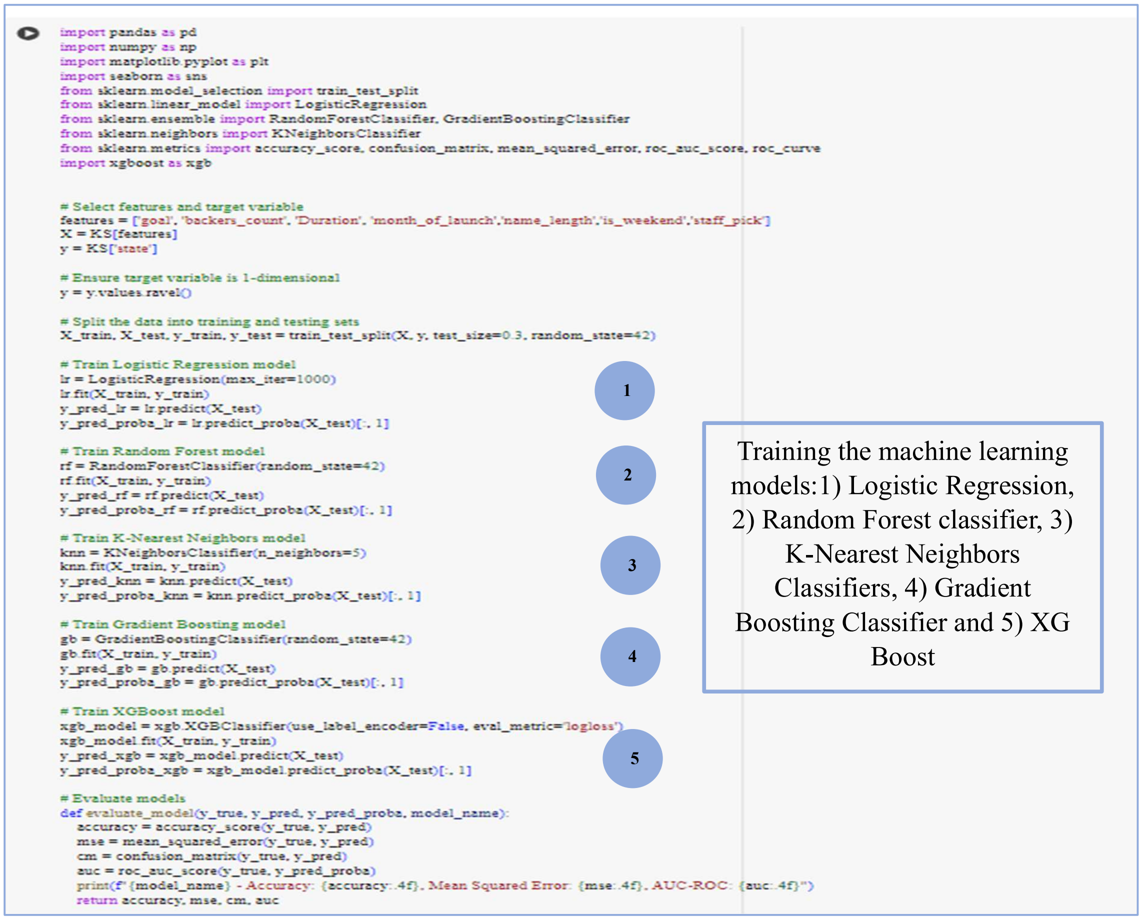
Analysing the blurb and name

```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk

# Ensure you have the NLTK data files
nltk.download('punkt')
nltk.download('stopwords')


# Fill NaN values with empty strings
KS['name'] = KS['name'].fillna('')

# Separate the data based on state
successful_projects = KS[KS['state'] == 1]
failed_projects = KS[KS['state'] == 0]

# Function to clean and tokenize text
def clean_and_tokenize(text):
    text = text.lower()
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word.isalpha()]
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    return tokens

# Analyze names for successful projects
successful_texts = successful_projects['name'].tolist()
successful_tokens = [clean_and_tokenize(text) for text in successful_texts]
successful_tokens_flat = [item for sublist in successful_tokens for item in sublist]
```


Failed Projects Word Cloud


Top Words in Successful Projects

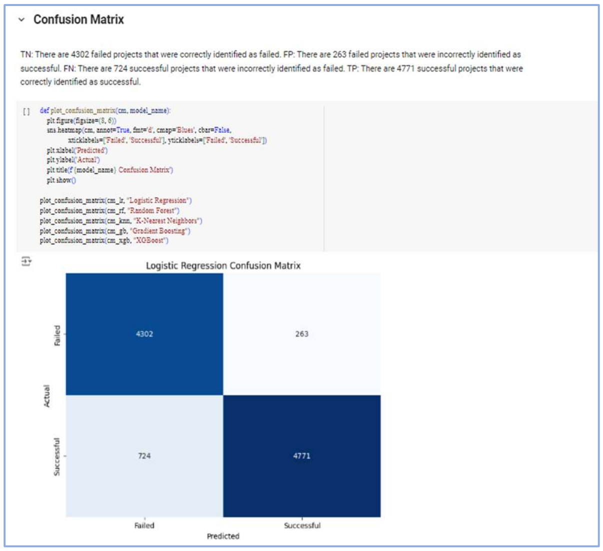
Modelling



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, mean_squared_error, roc_auc_score, roc_curve
import xgboost as xgb


# Select features and target variable
features = ['goal', 'backers_count', 'Duration', 'month_of_launch','name_length','is_weekend','staff_pick']
X = KS[features]
y = KS['state']

# Ensure target variable is 1-dimensional
y = y.values.ravel()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train Logistic Regression model
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
y_pred_proba_lr = lr.predict_proba(X_test)[:, 1]

# Train Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
y_pred_proba_rf = rf.predict_proba(X_test)[:, 1]

# Train K-Nearest Neighbors model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
y_pred_proba_knn = knn.predict_proba(X_test)[:, 1]

# Train Gradient Boosting model
gb = GradientBoostingClassifier(random_state=42)
gb.fit(X_train, y_train)
y_pred_gb = gb.predict(X_test)
y_pred_proba_gb = gb.predict_proba(X_test)[:, 1]

# Train XGBoost model
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
y_pred_proba_xgb = xgb_model.predict_proba(X_test)[:, 1]

# Evaluate models
def evaluate_model(y_true, y_pred, y_pred_proba, model_name):
   accuracy = accuracy_score(y_true, y_pred)
   mse = mean_squared_error(y_true, y_pred)
   cm = confusion_matrix(y_true, y_pred)
   auc = roc_auc_score(y_true, y_pred_proba)
   print(f"{model_name} - Accuracy: {accuracy:.4f}, Mean Squared Error: {mse:.4f}, AUC-ROC: {auc:.4f}")
   return accuracy, mse, cm, auc
```

Training the machine learning models:1) Logistic Regression, 2) Random Forest classifier, 3) K-Nearest Neighbors Classifiers, 4) Gradient Boosting Classifier and 5) XG Boost

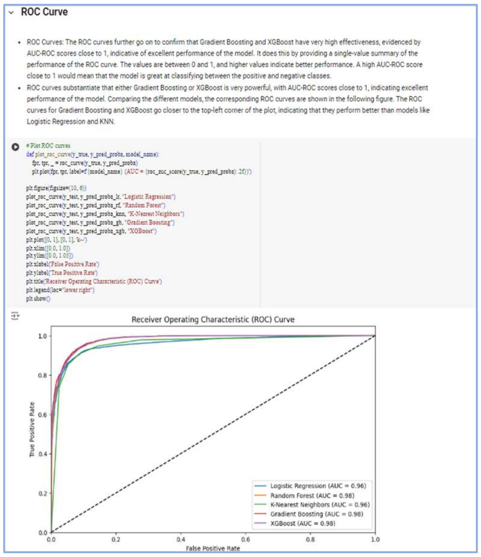
Evaluation



Confusion Matrix

TN: There are 4302 failed projects that were correctly identified as failed. FP: There are 263 failed projects that were incorrectly identified as successful. FN: There are 724 successful projects that were incorrectly identified as failed. TP: There are 4771 successful projects that were correctly identified as successful.

```
def plot_confusion_matrix(cm, model_name):
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Failed', 'Successful'], yticklabels=['Failed', 'Successful'])
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'{model_name} Confusion Matrix')
    plt.show()

plot_confusion_matrix(cm_lr, "Logistic Regression")
plot_confusion_matrix(cm_rf, "Random Forest")
plot_confusion_matrix(cm_knn, "K-Nearest Neighbors")
plot_confusion_matrix(cm_gb, "Gradient Boosting")
plot_confusion_matrix(cm_xgb, "XGBoost")
```

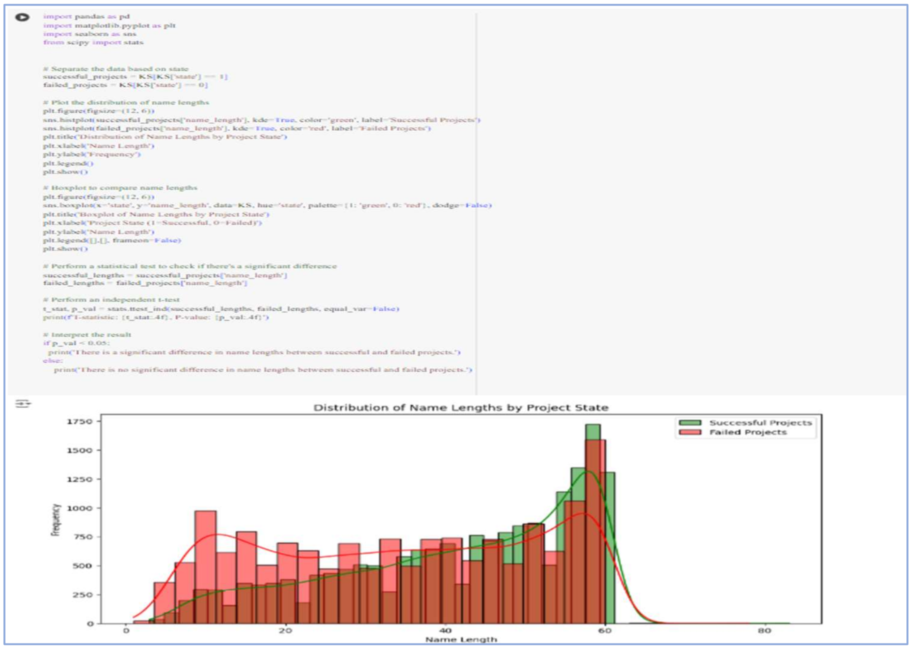


ROC Curve

- ROC Curves: The ROC curves further go on to confirm that Gradient Boosting and XGBoost have very high effectiveness, evidenced by AUC-ROC scores close to 1, indicative of excellent performance of the model. It does this by providing a single-value summary of the performance of the ROC curve. The values are between 0 and 1, and higher values indicate better performance. A high AUC-ROC score close to 1 would mean that the model is great at classifying between the positive and negative classes.
- ROC curves substantiate that either Gradient Boosting or XGBoost is very powerful, with AUC-ROC scores close to 1, indicating excellent performance of the model. Comparing the different models, the corresponding ROC curves are shown in the following figure. The ROC curves for Gradient Boosting and XGBoost go closer to the top-left corner of the plot, indicating that they perform better than models like Logistic Regression and KNN.

```
# Plot ROC curves
def plot_roc_curve(y_true, y_pred_proba, model_name):
    fpr, tpr, _ = roc_curve(y_true, y_pred_proba)
    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc_score(y_true, y_pred_proba):.2f})')

plt.figure(figsize=(10, 6))
plot_roc_curve(y_test, y_pred_proba_lr, "Logistic Regression")
plot_roc_curve(y_test, y_pred_proba_rf, "Random Forest")
plot_roc_curve(y_test, y_pred_proba_knn, "K-Nearest Neighbors")
plot_roc_curve(y_test, y_pred_proba_gb, "Gradient Boosting")
plot_roc_curve(y_test, y_pred_proba_xgb, "XGBoost")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

Receiver Operating Characteristic (ROC) Curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Logistic Regression (AUC = 0.96)
Random Forest (AUC = 0.98)
K-Nearest Neighbors (AUC = 0.96)
Gradient Boosting (AUC = 0.98)
XGBoost (AUC = 0.98)



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats


# Separate the data based on state
successful_projects = KS[KS['state'] == 1]
failed_projects = KS[KS['state'] == 0]

# Plot the distribution of name lengths
plt.figure(figsize=(12, 6))
sns.histplot(successful_projects['name_length'], kde=True, color='green', label='Successful Projects')
sns.histplot(failed_projects['name_length'], kde=True, color='red', label='Failed Projects')
plt.title('Distribution of Name Lengths by Project State')
plt.xlabel('Name Length')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Boxplot to compare name lengths
plt.figure(figsize=(12, 6))
sns.boxplot(x='state', y='name_length', data=KS, hue='state', palette={1: 'green', 0: 'red'}, dodge=False)
plt.title('Boxplot of Name Lengths by Project State')
plt.xlabel('Project State (1=Successful, 0=Failed)')
plt.ylabel('Name Length')
plt.legend([],[], frameon=False)
plt.show()

# Perform a statistical test to check if there's a significant difference
successful_lengths = successful_projects['name_length']
failed_lengths = failed_projects['name_length']

# Perform an independent t-test
t_stat, p_val = stats.ttest_ind(successful_lengths, failed_lengths, equal_var=False)
print(f"T-statistic: {t_stat:.4f}, P-value: {p_val:.4f}")

# Interpret the result
if p_val < 0.05:
  print("There is a significant difference in name lengths between successful and failed projects.")
else:
   print("There is no significant difference in name lengths between successful and failed projects.")
```