Predicting the news popularity in multiple social media platforms.

Team Member's

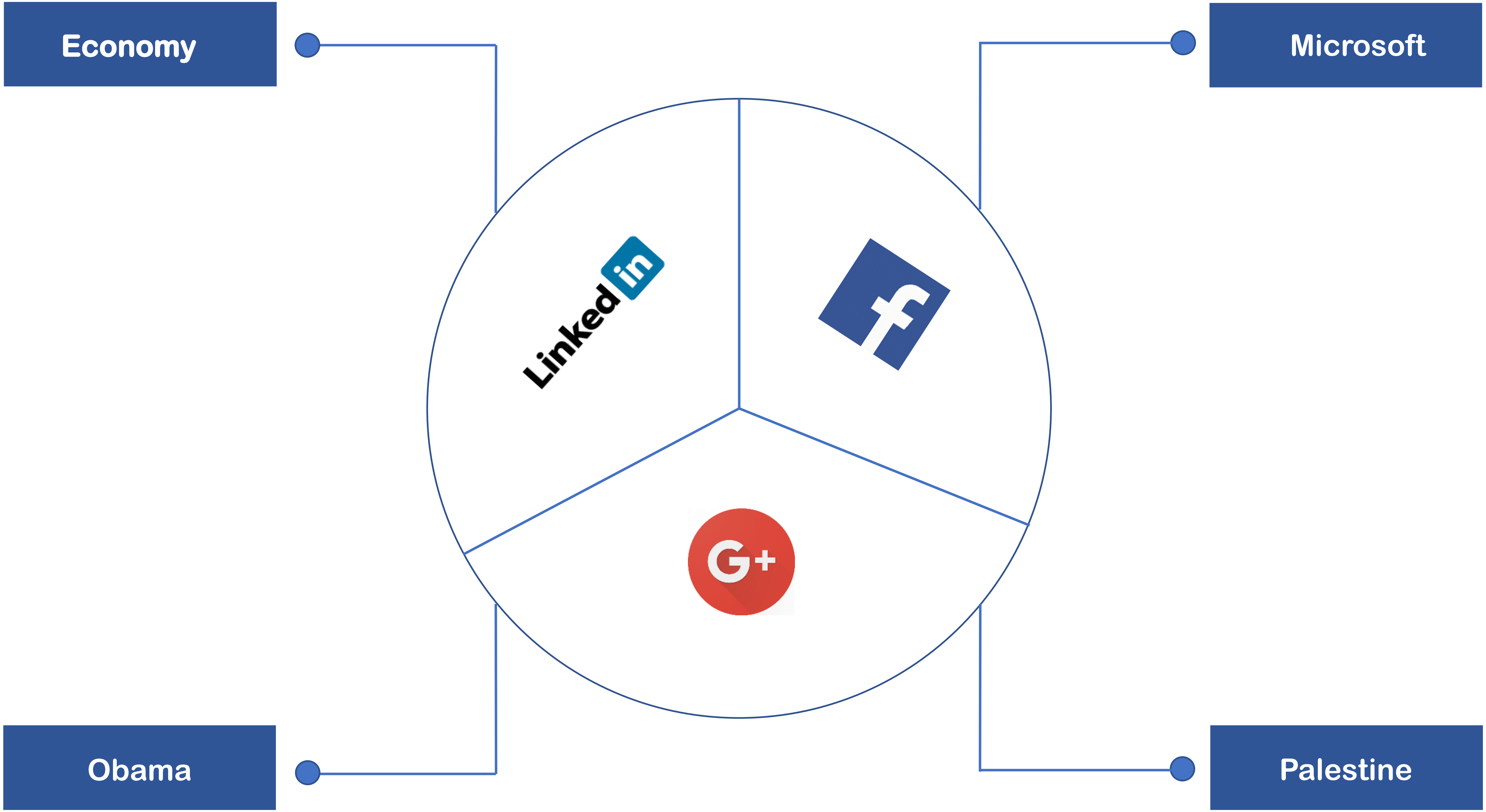
- Shivangi Patel
- Ravi Gohil
- Harsh Thakkar
- Kirti Pandey

Mentor Name

- Rohit Bhadauriya

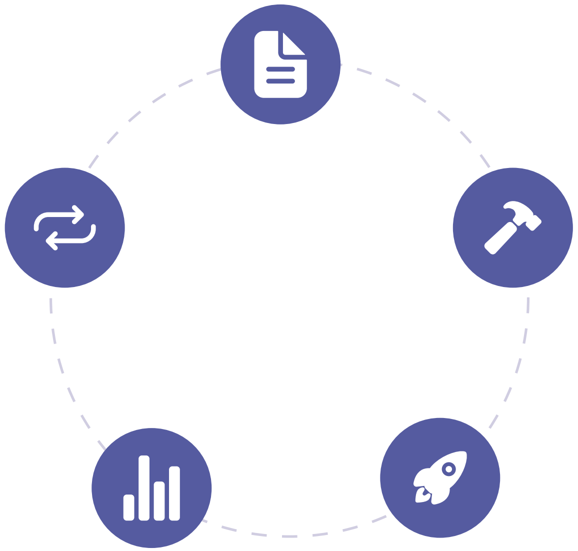
Publish Date

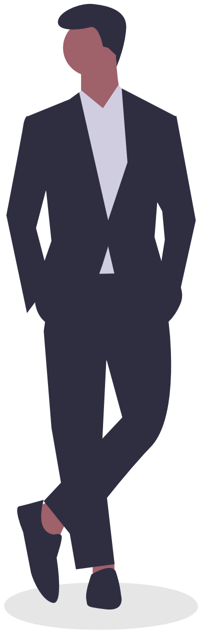
- 20-10-2022


Economy
Microsoft
Obama
Palestine

TEAM NEURON

ML(Machine Learning) Steps





1. Data overview
2. Data cleaning
3. Null value handling
4. EDA (Exploratory Data Analysis)
    a. Univariant
    b. Bivariant
    c. Multivariant
5. Feature Engineering
6. Create leaner regression Base model
7. VIF (Variance Inflation Factor)
8. Convert the target for classification
9. Applicated Under sampling
10. Create Base Model
11. Model Tuning
12. Cross validation (grid search CV)
13. Applied Oversampling
14. Base Model create
15. Hyperparameter Tuning
16. Final model

1) Data Overview

TEAM NEURON

Data that has been interpreted and manipulated and has now some meaningful inference for the users.

1. Shape of the data

In [3]:
```
df_newsone.shape
```
Out[3]:
```
(93239, 11)
```

2. Check the columns(features)

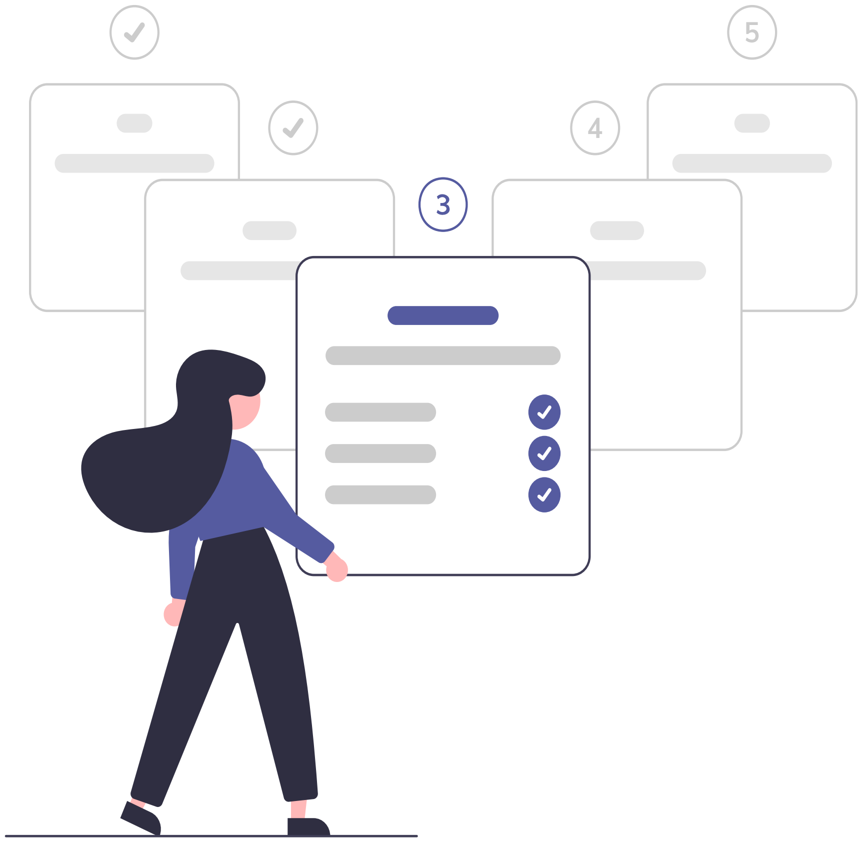
In [4]:
```
df_newsone.columns
```
Out[4]:
```
Index(['IDLink', 'Title', 'Headline', 'Source', 'Topic', 'PublishDate',
       'SentimentTitle', 'SentimentHeadline', 'Facebook', 'GooglePlus',
       'LinkedIn'],
      dtype='object')
```

3. Describe the dataset

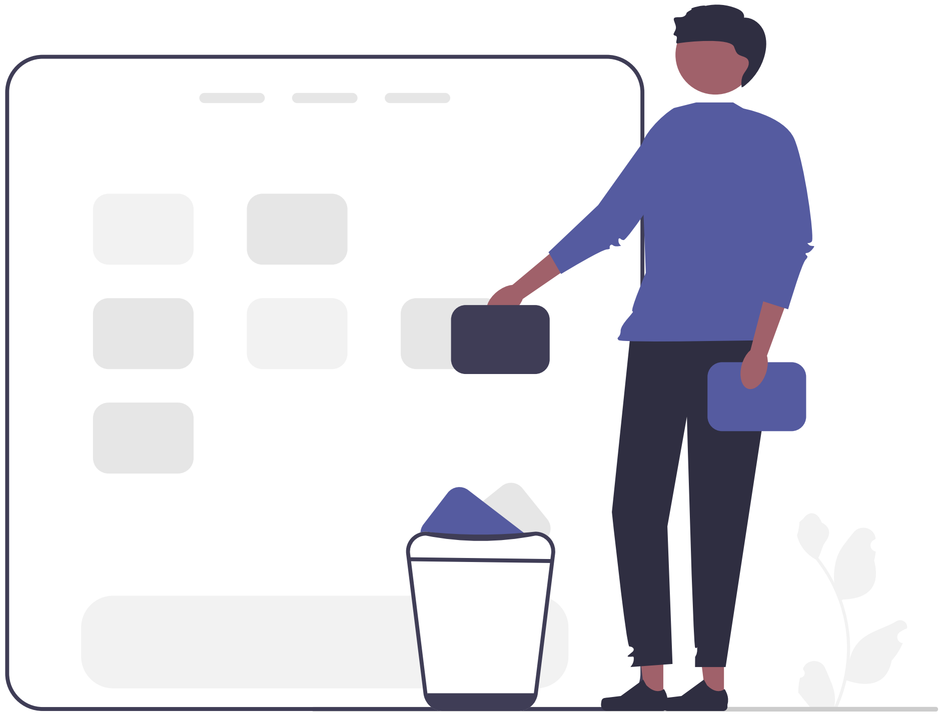
In [5]:
```
df_newsone.describe(include='all').T
```
Out[5]:

4. Check for data types

In [6]:
```
df_newsone.info()
```



2) Data cleaning



Data Cleaning machine learning is the method of identifying the incomplete, wrong, unnecessary, incorrect, or missing part of the data and then changing, replacing, or removing them according to the specific requirement.

- Change a data type

In [8]:

```
df_newsone.IDLink= df_newsone.IDLink.astype('int')
```

TEAM NEURON

3) Null value handling

Null values can take on many different forms in machine learning, such as missing data, invalid data, or incorrect data.

1. Check percentage of null values

In [9]:

```
(df_newsone.isnull().sum()*100)/len(df_newsone)
```

Out[9]:

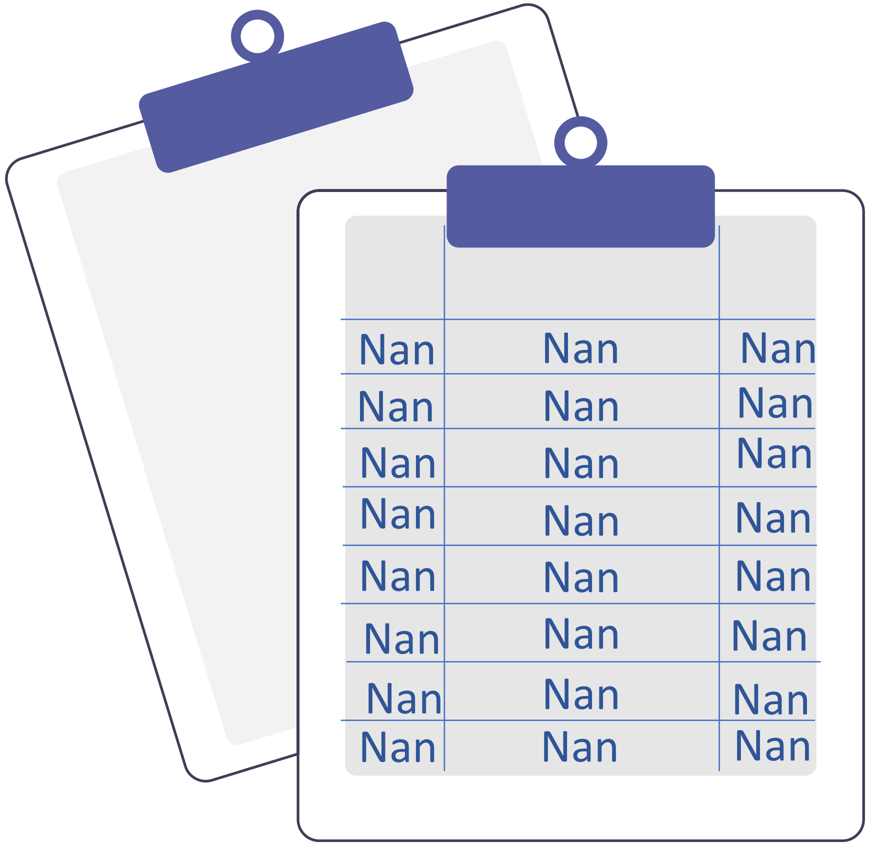
```
IDLink              0.000000
Title               0.000000
Headline            0.016088
Source              0.299231
Topic               0.000000
PublishDate         0.000000
SentimentTitle      0.000000
SentimentHeadline   0.000000
Facebook            0.000000
GooglePlus          0.000000
LinkedIn            0.000000
dtype: float64
```

- there is 0.29% null values in `Source` so we drop that rows
- and `Headline` columns 0.016% null values

2. Deal with null values

In [10]:

```
df_newsone=df_newsone.dropna().copy()
```



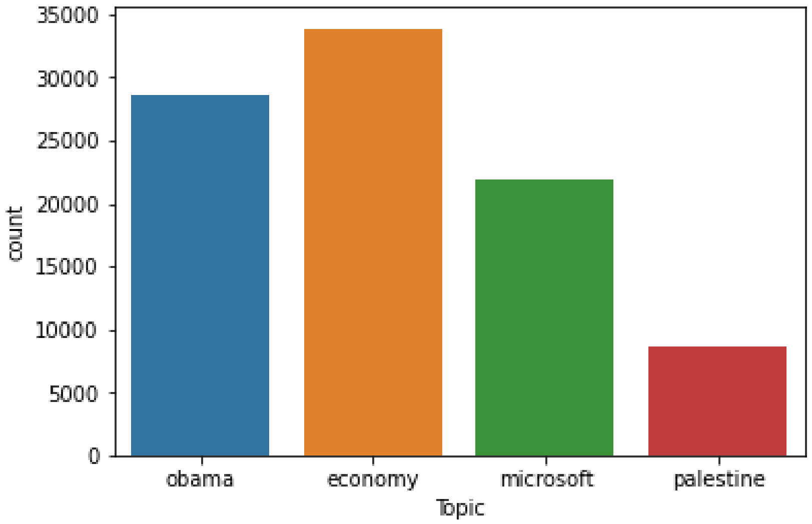
TEAM NEURON

4) EDA (Exploratory Data Analysis)

A method for summarizing data, identifying patterns and relationships, and detecting outliers is exploratory data analysis. This type of data analysis is most often used when the data set is large or complex, and it can help with data comprehension.

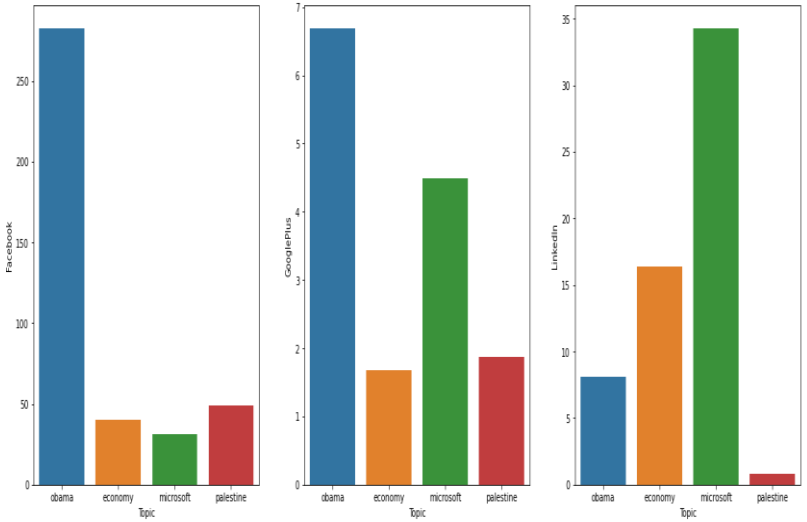
Univariate

Univariate function optimization involves finding the input to a function that results in the optimal output from an objective function. This is a common procedure in machine learning when fitting a model with one parameter or tuning a model that has a single hyperparameter.



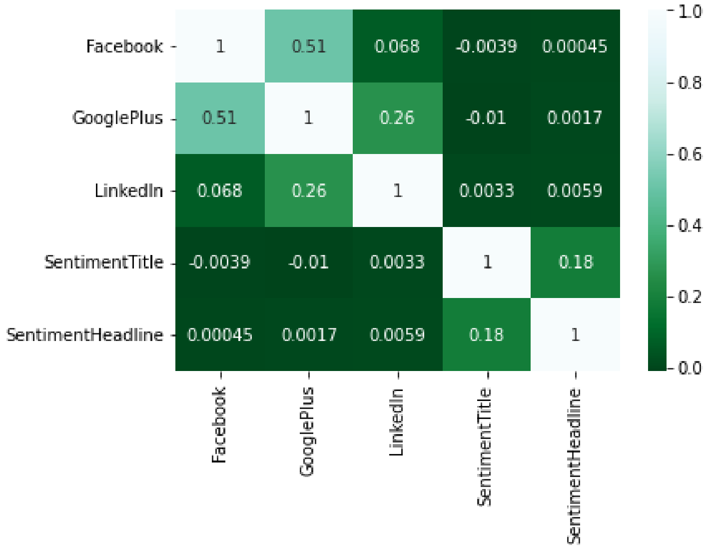
Bivariate

Two variables are involved in bivariate data. Bivariate analysis is concerned with causes and relationships, and the goal is to determine the relationship between the two variables.



Multivariate

Multivariate classification is a machine learning technique used to predict the class of an observation based on multiple features or variables. It is a form of supervised learning, meaning that it relies on labeled training data to learn the relationship between the features and the classes.



4.1) EDA Summary

- . Through our analysis we could figure that amongst all the four news topics, news items related to Economy were most popular on social media.
- 2. Secondly, we figured that Facebook has higher reach as compared to GooglePlus and LinkedIn.
- 3. Next we found out that the news items related to the topic Obama trended more on Facebook and Google plus. Also, the news items related to Microsoft trended more on LinkedIn.
- 4. In multivariate analysis we could conclude that 51% of Facebook news are also shared on Google plus and vice-versa.

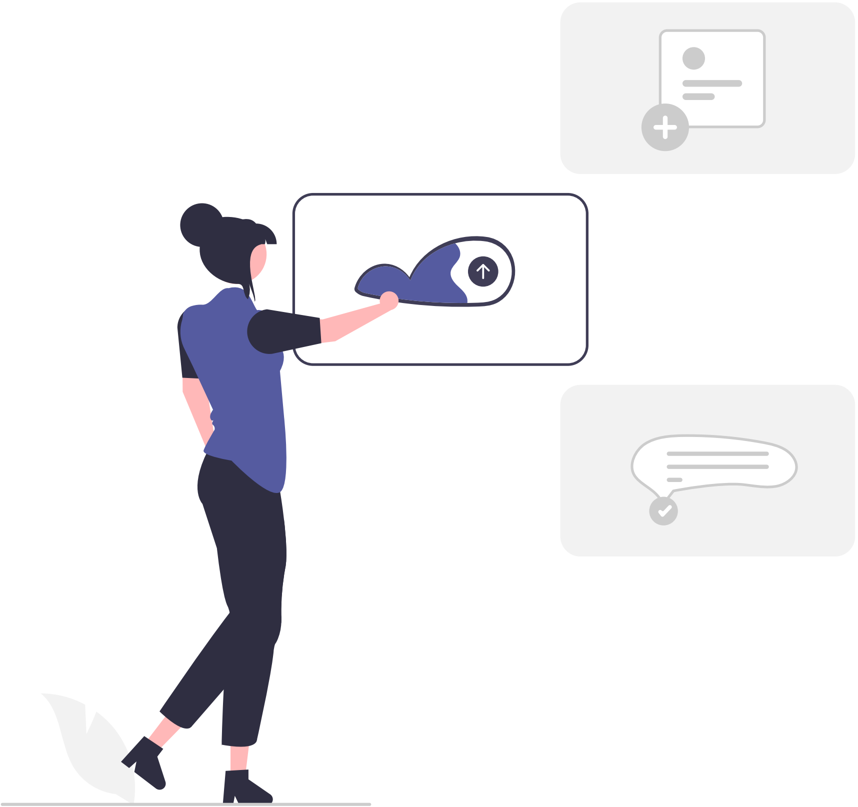
5) Feature Engineering

Feature engineering or feature extraction or feature discovery is the process of using domain knowledge to extract features from raw data.

1.Featureselction

2.Feature extraction

3.Encoding And Scaling



Feature selection

In feature selection we select only most relevant column over the dataset .

In [24]:

```
df_news.drop('PublishDate',axis=1,inplace=True)
```

In [25]:

```
df_news_copy=df_news.copy()
target_variable =  list(df_news_copy["SentimentTitle"].copy())
df_news_copy.drop("SentimentTitle",axis=1,inplace=True)
```

Feature extraction

Feature extraction is very different from Feature selection: the former consists in **transforming arbitrary data, such as text or images, into numerical features usable for machine learning.**

```
# Split the values according to our requirement
# Separate the Time
df_news['Publish_Time']=df_news['PublishDate'].str.split(" ").str[1]
# Separate the date
df_news['Publish_Date']=df_news['PublishDate'].str.split(" ").str[0]
# convert for datatype
df_news['Publish_Date']= pd.to_datetime(df_news['Publish_Date'])

# Separate the month
df_news['Publish_Month']= df_news['Publish_Date'].dt.month
# Separate the day
df_news['Publish_Day']= df_news['Publish_Date'].dt.day
# Map the Month in terms of words
df_news["Season"]=df_news["Publish_Month"].copy()
df_news.Season.replace({1:'Winter',2:'Winter',3:'Spring',4:'Spring',5:'Spring',6:'Summer',7

df_news.Publish_Month.replace({1:'January',2:'February',3:'March',4:'April',5:'May',6:'June

# Print the first 5 record
df_news.head()
```

Encoding And Scaling

Encoding and scaling is technique which use data is ready for machine model . And model is work on good manner .

In [27]:

```
# Do LableEncoding
le = preprocessing.LabelEncoder()
df_news_copy_object_enco = df_news_copy_object.apply(le.fit_transform)
df_news_copy_object_enco.head()
```

In [29]:

```
# Use StandScaler for scaling
X_scaler = StandardScaler()
num_scaler = X_scaler.fit_transform(concat_news_dataframe)
X=pd.DataFrame(num_scaler,columns=concat_news_dataframe.columns)
```

TEAM NEURON

6) leaner regression Base model

Linear Regression is **a machine learning algorithm based on supervised learning**. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

In [32]:

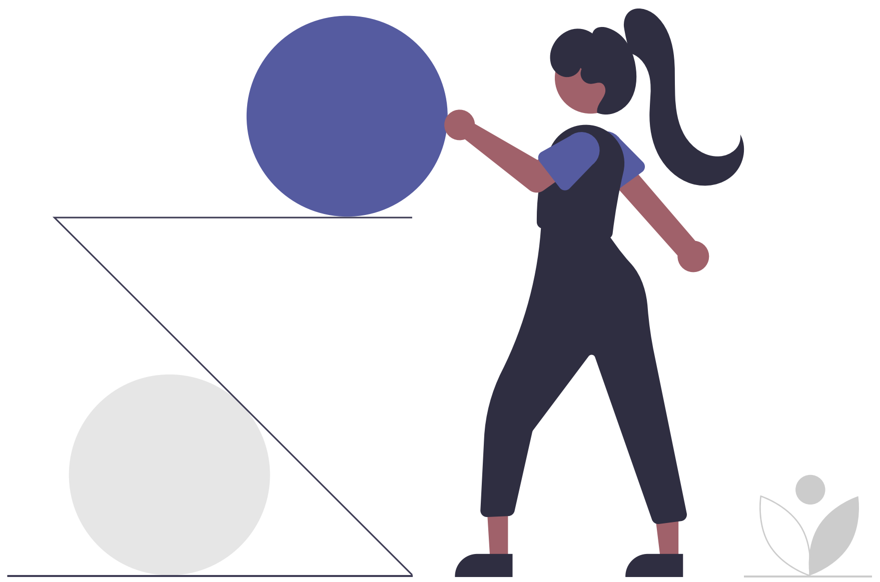
```
# Use Linear Regression As a base model
base_model = LinearRegression().fit(X_train,Y_train)
```

In [33]:

```
# Check the score for base model
base_model.score(X_train,Y_train)
```

Out[33]:

```
0.03357131449689199
```



TEAM NEURON

7) VIF (Variance Inflation Factor)

A variance inflation factor (VIF) is **a measure of the amount of multicollinearity in regression analysis**. Multicollinearity exists when there is a correlation between multiple independent variables in a multiple regression model. This can adversely affect the regression results.

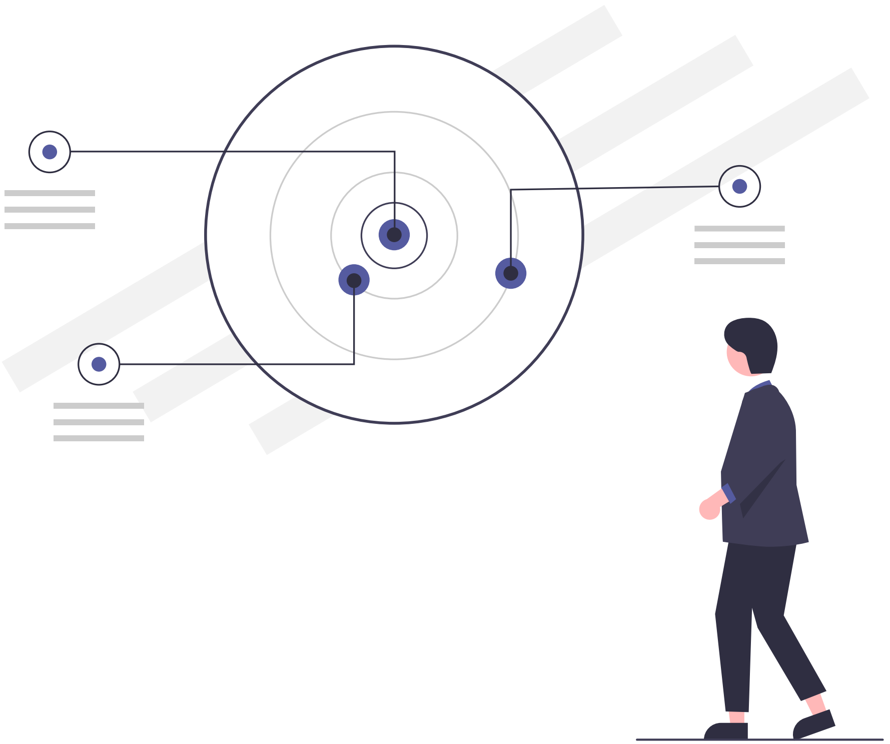
```
for ind in range(len(df_news_VIF_number.columns)):
    vif = pd.DataFrame()
    vif["VIF_Factor"] = [variance_inflation_factor(df_news_VIF_number,i) for i in range(df_
    vif["Features"]=df_news_VIF_number.columns
    multi =  vif[vif['VIF_Factor']>10]


    if (multi.empty == False):
        df_sorted = multi.sort_values(by = 'VIF_Factor',ascending= False)
    else:
        print(vif)
        break


    if (df_sorted.empty == False):
        df_features_vif = df_news_VIF_number.drop(df_sorted.Features.iloc[0], axis=1)
    else:
        print(vif)
        break
```

```
   VIF_Factor           Features
0    2.363040             IDLink
1    1.036659     SentimentTitle
2    1.067113  SentimentHeadline
3    1.416700           Facebook
4    1.515392         GooglePlus
5    1.093871           LinkedIn
6    2.338872        Publish_Day
```

8) Convert the target for classification

We are use np.round function and we get three category 1,-1,0

- -1 not influence by people.
- 0 moderate influence by people.
- 1 highly influence by people.



In [42]:

```
df_news["target_sentiment"] = np.round(df_news["SentimentTitle"])
```

In [43]:

```
df_news.target_sentiment.value_counts()
```

Out[43]:

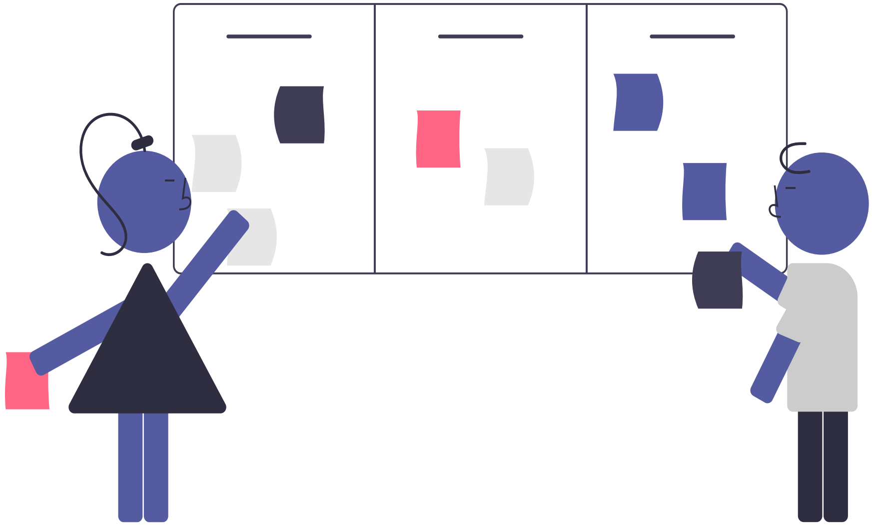
```
 0.0    92640
-1.0      172
 1.0      133
Name: target_sentiment, dtype: int64
```

TEAM NEURON

9) Applicated Under sampling

If our data is imbalance at that time we perform over sampling and under sampling .

Here our target variable is imbalance so we apply under sampling on target variable column . For balancing a categorical values.



```
from imblearn.under_sampling import RandomUnderSampler

rus =  RandomUnderSampler(random_state=0)
X_resampled,y_resampled = rus.fit_resample(X1,Y1)
print(sorted(Counter(y_resampled).items()),y_resampled.shape)
```

```
[(-1.0, 133), (0.0, 133), (1.0, 133)] (399,)
```

TEAM NEURON

10) Apply Decision Tree

- **First under sample based model :**

In [69]:

```
# Classification report for traning data
print(classification_report(y_train_under,predict_train))
```

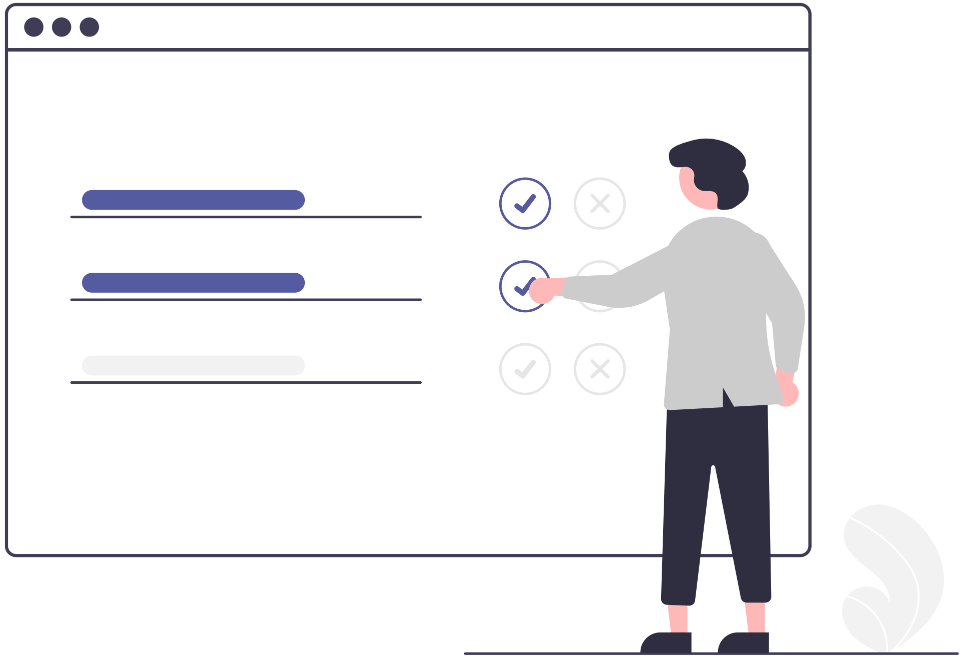
```
              precision    recall  f1-score   support

        -1.0       1.00      1.00      1.00        95
         0.0       1.00      1.00      1.00        96
         1.0       1.00      1.00      1.00        88

    accuracy                           1.00       279
   macro avg       1.00      1.00      1.00       279
weighted avg       1.00      1.00      1.00       279
```

In [70]:

```
# Classification report for testing data
print(classification_report(y_test_under,predict_test))
```

```
              precision    recall  f1-score   support

        -1.0       1.00      1.00      1.00        38
        -0.0       1.00      0.97      0.99        37
         1.0       0.98      1.00      0.99        45

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

TEAM NEURON



- **Model tuning :**

  - The goal of model tuning is to optimize the values of Hyperparameters. Major characteristics of hyperparameters are – These are tuned They are external values These are defined by a user These aren’t part of a trained model. Hyperparameter optimization is another term for model tuning.

In [73]:

```
# Set the hyperparameter
tuned_params=[{'criterion':['entopy','gini'],'max_depth':[10,20,30],'max_features':['log2',
```

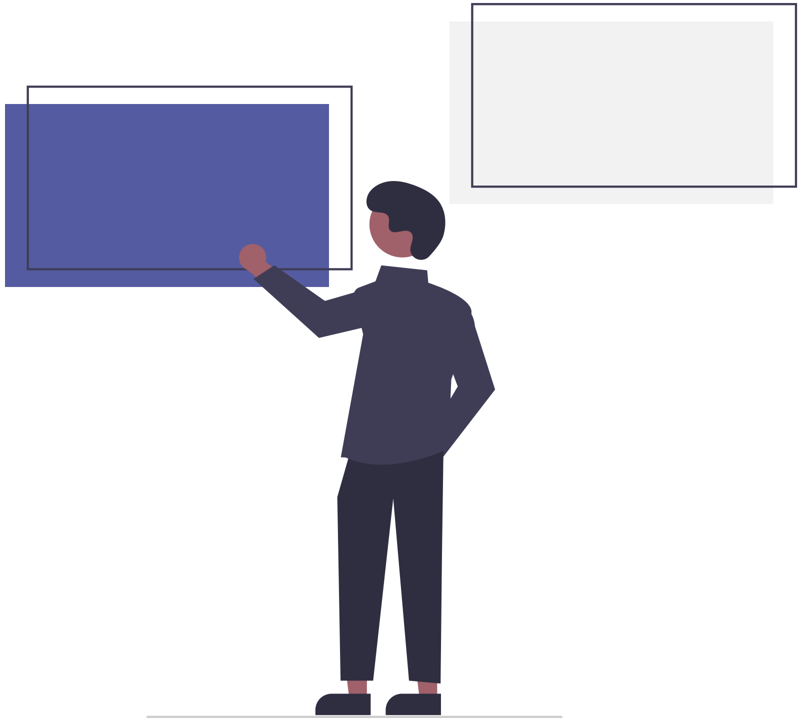
TEAM NEURON

- **Cross validation (grid search CV)**

GridSearchCV is **a technique to search through the best parameter values from the given set of the grid of parameters .**

```
# Find the best hyperparameter using grid serch cv
over_tree_grid=GridSearchCV(estimator=decision_tree_clasifier,param_grid=tuned_params,cv=5)
model = over_tree_grid.fit(x_train_under,y_train_under)
print(model.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 10, 'max_features': 'log2', 'max_leaf_nod
es': 50, 'min_samples_leaf': 1, 'min_samples_split': 11}
```



TEAM NEURON

Gride search cv is return good hyperparameter . And that parameter we use for creating new model over the under sample data set .

In [78]:

```
# Classification report for traing data
print(classification_report(y_train_under,predicted_train))
```

	precision	recall	f1-score	support
-1.0	0.87	0.92	0.89	95
0.0	0.92	0.94	0.93	96
1.0	1.00	0.92	0.96	88
accuracy			0.92	279
macro avg	0.93	0.92	0.93	279
weighted avg	0.93	0.92	0.93	279

In [79]:

```
#  Classification report for testing data
print(classification_report(y_test_under,predicted_test))
```

	precision	recall	f1-score	support
-1.0	0.62	0.66	0.64	38
-0.0	0.56	0.65	0.60	37
1.0	0.78	0.64	0.71	45
accuracy			0.65	120
macro avg	0.66	0.65	0.65	120
weighted avg	0.66	0.65	0.65	120

As we can see model is overfitted, the reason is taring accuracy is 92% and testing accuracy 65%

TEAM NEURON

We also do same process for over sampling

But In over sampling data set is work good with Decision Tree model

Final Model

In [102]:

```
print(classification_report(y_train_over,predict_train_over))
```

	precision	recall	f1-score	support
-1.0	1.00	1.00	1.00	64997
0.0	1.00	1.00	1.00	64700
1.0	1.00	1.00	1.00	64847
accuracy			1.00	194544
macro avg	1.00	1.00	1.00	194544
weighted avg	1.00	1.00	1.00	194544

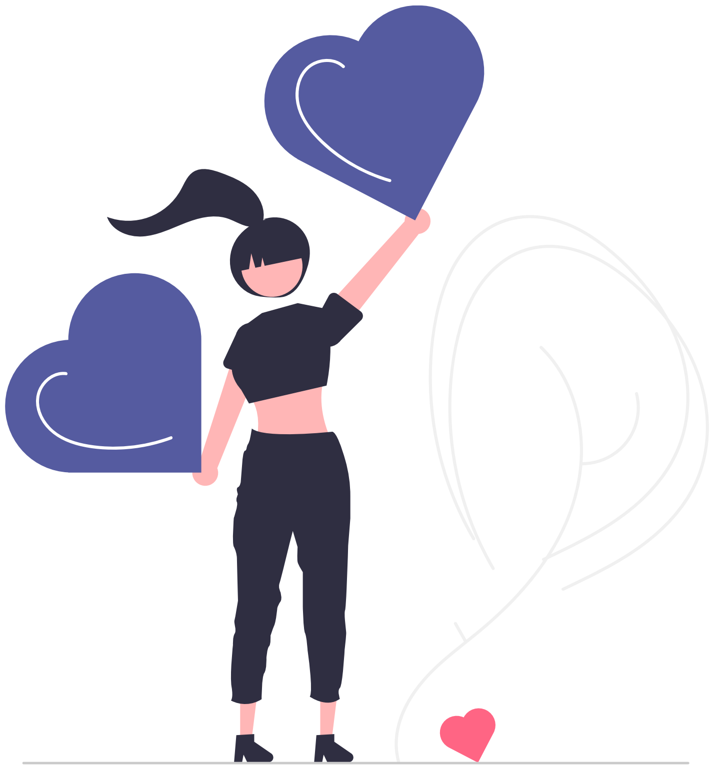
In [103]:

```
print(classification_report(y_test_over,predict_test_over))
```

	precision	recall	f1-score	support
-1.0	1.00	1.00	1.00	27643
0.0	1.00	1.00	1.00	27940
1.0	1.00	1.00	1.00	27793
accuracy			1.00	83376
macro avg	1.00	1.00	1.00	83376
weighted avg	1.00	1.00	1.00	83376

In [104]:

As we can see here oversampling concept good fit for the data That proven by seen the accuracy of taring and testing.



Thank you !