**Detailed Workflow — YouTube Video Views Prediction & Viral Classification**

1. Data Collection:

* Pulled data via YouTube Data API v3
* Saved key metrics per video (views, likes, comments, published\_date, channel\_title, etc.)

**Output of this step**:  
A dataset where each row = 1 video with snapshot metrics.

1. Data Cleaning

* Ensure no missing values in critical columns (views, likes, comments, published\_date)
* Convert published\_date to datetime

1. Feature Engineering:

* **From date**: day, month, year, quarter, week\_of\_year, day\_of\_month, is\_weekend
* **From engagement**: like\_ratio, comment\_ratio, views\_per\_comment, likes\_per\_comment

Encode categorical features:

* One-hot encode day
* One-hot encode channel\_title (if used)

1. Target Definition:

**Regression**

* Target = views (or log\_views if distribution is skewed)

**Classification**

* Define viral video:

a threshold that makes business sense (e.g. >100K views)

**Output**:

* A numeric target for regression
* A binary target for classification

1. Train-Test Split:

* Split dataset into train/test sets (e.g. 80% train / 20% test)
* Ensure both splits representatively cover channels and date ranges

**Output**:

* X\_train, y\_train\_reg (for regression)
* X\_train, y\_train\_cls (for classification)
* X\_test, y\_test\_reg, y\_test\_cls

1. Model Building:

**Regression model**

* Baseline: Gradient Boosting Regressor : XGBoost

**Why?**

* Handles skewed data + nonlinear relationships better than Linear Regression
* Works well on medium-sized datasets

Evaluate:

* MAE
* RMSE
* R²
* Plot actual vs predicted

**Classification model**

* Baseline: Gradient Boosting Classifier :XGBoost

**Why?**

* Handles class imbalance better (can set scale\_pos\_weight in XGBoost)

Evaluate:

* Accuracy
* Precision, Recall, F1-score
* ROC-AUC
* Confusion matrix