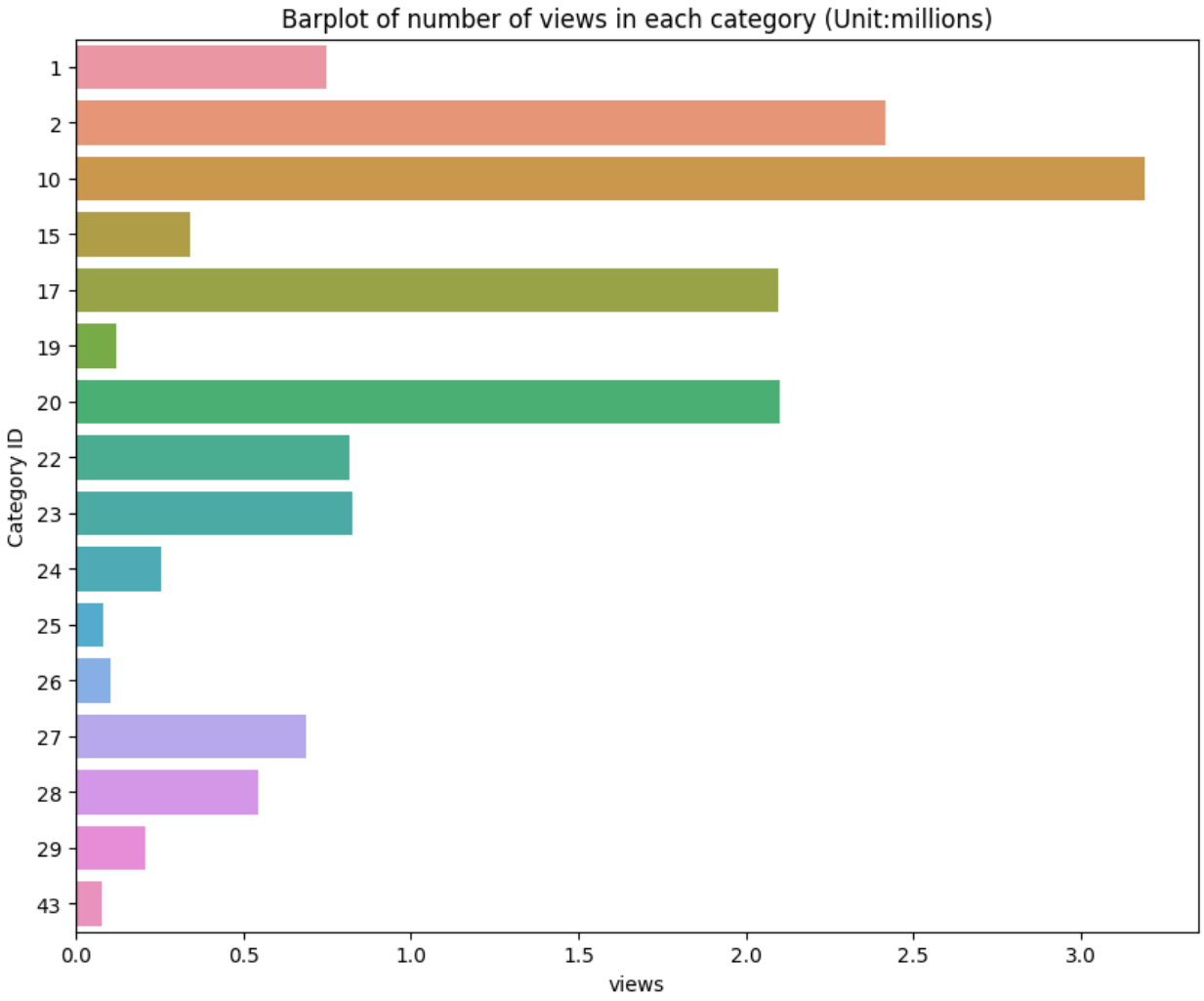
```
import numpy as np
import pandas as pd

numerics = ['int8','int16', 'int32', 'int64', 'float16', 'float32',
'float64']

data = pd.read_csv('/content/drive/MyDrive/kaggle/input/USvideos.csv',
sep=',')

#dataset size
len(data)
data.info()



##Prepare data type columns
data['trending_date'] = pd.to_datetime(data['trending_date'],
format='%y.%d.%m')
data['publish_time'] = pd.to_datetime(data['publish_time'],
infer_datetime_format=True)

##Adding a new columns for publish date and time
data['publish_date'] = data['publish_time'].dt.date
data['publish_wd'] = data['publish_time'].dt.weekday
data['publish_hr'] = data['publish_time'].dt.hour
data['publish_time'] = data['publish_time'].dt.time

#remove columns not useful for training the model
data = data.drop(['tags', 'video_error_or_removed',
'description'],axis = 1)

#dropping duplicates, keeping the first value
dsta = data.drop_duplicates(keep = 'first')

data.info()
data.head()

import seaborn as sb
import matplotlib.pyplot as plt
dff = data[['category_id',
'views']].groupby('category_id').aggregate(np.sum).reset_index()\
.sort_values(by='views', ascending=False)
dff.views = data.views/10**6
plt.figure(figsize=(10,8))
view_box = sb.barplot(x='views', y='category_id',data=dff, orient='h')
plt.title('Barplot of number of views in each category
(Unit:millions)')
plt.ylabel('Category ID')
plt.xlabel('views')
```

```
#Education, Film&Animation, and comedy are what Americans watch the most
```

```
Text(0.5, 0, 'views')
```



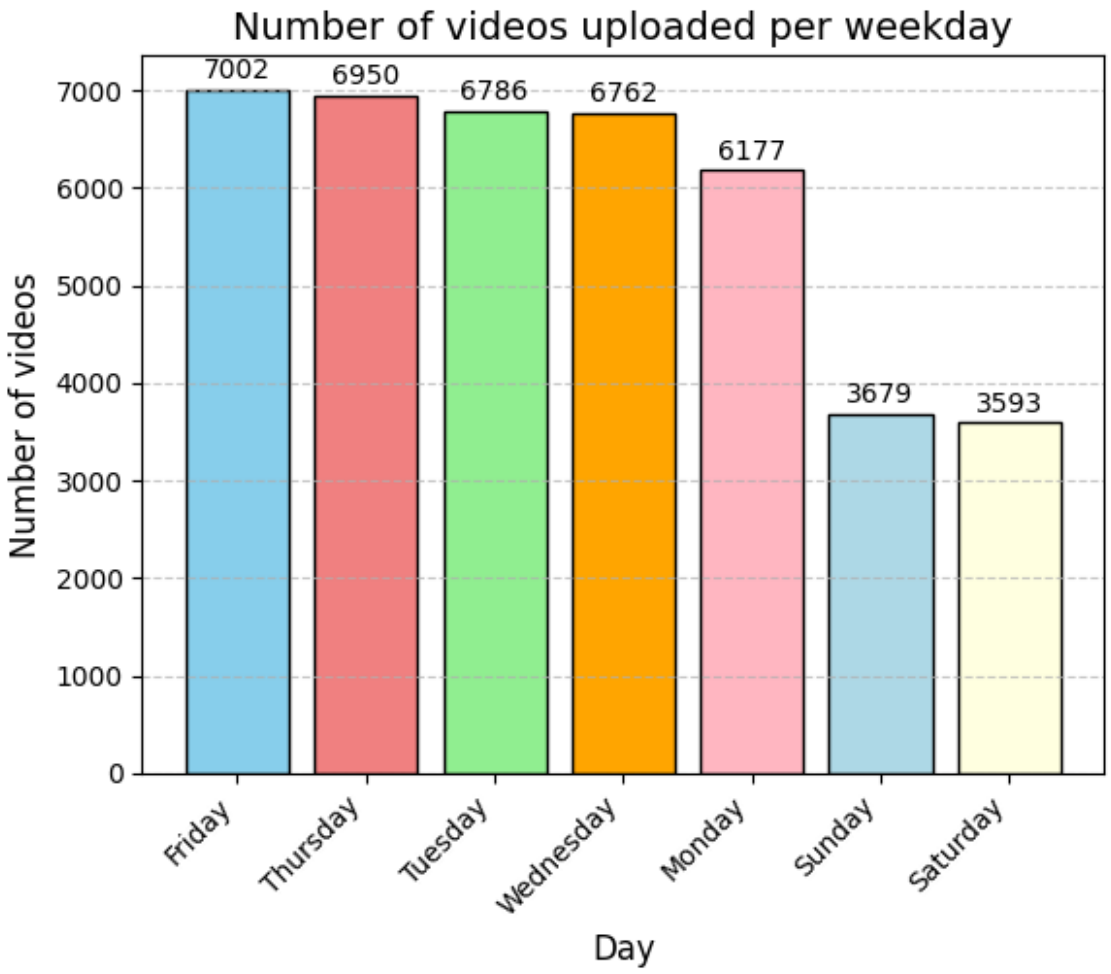

```
print(data[['views', 'likes']].corr())
print(data[['views', 'dislikes']].corr())
```

```
          views     likes
views  1.000000  0.849177
likes  0.849177  1.000000
             views  dislikes
views     1.000000  0.472213
dislikes  0.472213  1.000000
```

What we discovered:

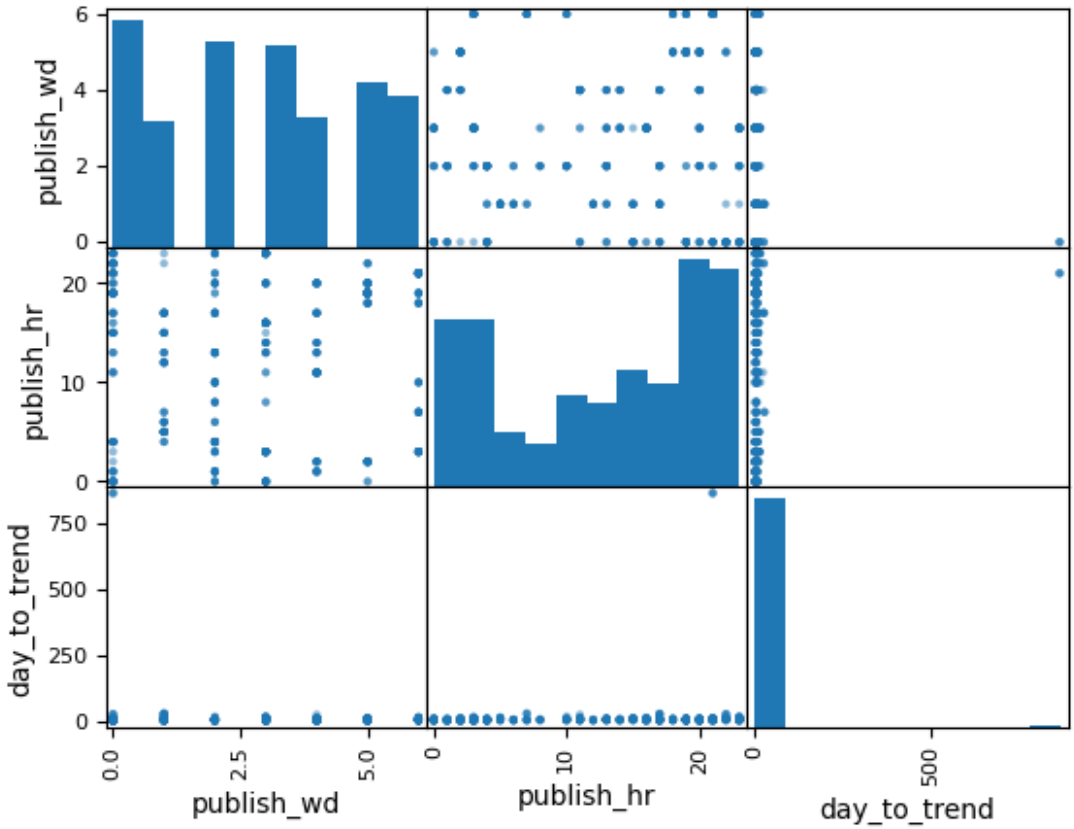
1. The correlation between the view and like count is very high 0.85. Which means if the view count is high then the number of likes is highly influenced by it too.
2. The correlation between view count and dislike count is 0.47, meaning, the dislike count is also highly influenced by the number of views. Or in simpler words, to make a video popular, it does not require high content quality/positive reactions(high like count).

```
# Date and time analysis

data_with_days = data['publish_wd'].map(dict(zip(range(7),

['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday
'])))).value_counts()

import matplotlib.pyplot as plt

colors = ['skyblue', 'lightcoral', 'lightgreen', 'orange',
'lightpink', 'lightblue', 'lightyellow']
plt.bar(data_with_days.index.values, data_with_days, color=colors,
edgecolor='black')

plt.xlabel('Day', fontsize=12)
plt.ylabel('Number of videos', fontsize=12)
plt.title('Number of videos uploaded per weekday', fontsize=14)

plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=45, ha='right')

for i, value in enumerate(data_with_days):
    plt.text(i, value + 50, str(value), ha='center', va='bottom',
fontsize=10)

plt.show()
```




```
#number of days it takes for a video to get to trending page on
youtube

#considering only videos with disabled comments to reduce training
time and also because comment count is currently not useful
new_data = data.loc[(data.comments_disabled) &
(~data.ratings_disabled)].copy()
#Create a new column for the number of days a video takes to get on
the trending list
new_data['day_to_trend'] =
abs(np.subtract(new_data.trending_date.dt.date,new_data.publish_date).
apply(lambda x: x.days))
left_vars =
['views','likes','dislikes','comment_count','publish_wd','publish_hr',
'day_to_trend','title']

new_data = new_data[left_vars]
new_data.reset_index(inplace=True)
new_data.head()
new_data.info()
```

```
from pandas.plotting import scatter_matrix
scatter_matrix(new_data[['publish_wd', 'publish_hr', 'day_to_trend']])
plt.show()
```



no normal distribution of numerical values -> Random forest model

```
#RANDOM FOREST

import sklearn
from sklearn.model_selection import cross_val_score, GridSearchCV,
train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
#considering only videos which took less than a week to get on
trending
new_data.day_to_trend = new_data.day_to_trend <= 7

X = new_data[['views', 'likes', 'dislikes', 'publish_wd',
'publish_hr']]
y = new_data['day_to_trend']

X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.2,
random_state=4)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp,
test_size=0.25, random_state=4)


#grid-search (1)
gsc = GridSearchCV(
        estimator=RandomForestClassifier(),
        param_grid = {'max_depth': range(6,10),'n_estimators':
range(155,170)},cv=5, scoring='accuracy', verbose=1, n_jobs=-1)
grid_result = gsc.fit(X,y)

print(grid_result.best_params_,grid_result.best_score_)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
{'max_depth': 6, 'n_estimators': 155} 0.6886612758310872
```

```
from sklearn.metrics import accuracy_score

rfc = RandomForestClassifier(oob_score=True, warm_start=False)

param_grid = {'max_depth': range(6,10), 'n_estimators':
range(155,170)}
best_score = 0

for max_depth in param_grid['max_depth']:
    for n_estimators in param_grid['n_estimators']:
        rfc.set_params(max_depth=max_depth, n_estimators=n_estimators)

        rfc.fit(X_train, y_train)

        val_predictions = rfc.predict(X_val)

        val_score = accuracy_score(y_val, val_predictions)

        if val_score > best_score:
            best_score = val_score
            best_params = {'max_depth': max_depth, 'n_estimators':
n_estimators}

print(best_params, best_score)

rfc.set_params(**best_params)
rfc.fit(X_temp, y_temp)

test_predictions = rfc.predict(X_test)
test_score = accuracy_score(y_test, test_predictions)
print("Test Score: ", test_score)
```

```
{'max_depth': 8, 'n_estimators': 159} 0.8962264150943396
Test Score:  0.8962264150943396
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score

test_predictions = rfc.predict(X_test)

accuracy = accuracy_score(y_test, test_predictions)
print("Accuracy: ", accuracy)

f1 = f1_score(y_test, test_predictions, average='weighted')
print("F1 Score: ", f1)

precision = precision_score(y_test, test_predictions,
average='weighted')
print("Precision: ", precision)

recall = recall_score(y_test, test_predictions, average='weighted')
print("Recall: ", recall)

print("OOB Score: ", rfc.oob_score_)

from sklearn.metrics import classification_report

test_predictions = rfc.predict(X_test)

print("\nClassification Report:\n")
print(classification_report(y_test, test_predictions))

score = metrics.f1_score(np.array(y_test),test_predictions)
print("f1 score: ",score)
print("Accuracy:", metrics.accuracy_score(y_test, test_predictions))

def personalized_prediction():
  views = input("Number of views for the video currently: ")
  likes = input("Number of likes for the video currently: ")
  dislikes = input("Number of dislikes for the video currently: ")
  day = input("day of week (0 for sunday - 6 for saturday) :")
  hour = input("hour uploaded (0 to 23, in 24hr format) :")
  test_data =
pd.DataFrame([[views,likes,dislikes,day,hour]],columns=['views',
'likes', 'dislikes', 'publish_wd', 'publish_hr'])
  prediction = rfc.predict(test_data)
  print("Is there a chance of the video getting on the trending page
on youtube? : ", 'yes' if prediction[0] else 'no')
  print()

res = [personalized_prediction() for _ in range(2)]
```

```
Number of views for the video currently: 89
Number of likes for the video currently: 9
Number of dislikes for the video currently: 2
day of week (0 for sunday - 6 for saturday) :2
```

```
hour uploaded (0 to 23, in 24hr format) :2
Is there a chance of the video getting on the trending page on
youtube? :  yes

Number of views for the video currently: 12
Number of likes for the video currently: 4
Number of dislikes for the video currently: 2
day of week (0 for sunday - 6 for saturday) :2
hour uploaded (0 to 23, in 24hr format) :4
Is there a chance of the video getting on the trending page on
youtube? :  yes
```