IBM 312

DATA MINING FOR BUSINESS INTELLIGENCE

Report and Analysis on Youtube Views Prediction

Team Members:

- Ashish Singh (20310007)
- Pawan
- Rishabh Kumar
- Chirag Baghasingh

Problem Statement

The problem aims to develop a predictive linear model that estimates the number of views for trending videos based on factors such as likes, dislikes, comments, and other variables. The objective is to identify the statistically significant factors influencing the view count and assess the relationship between the number of comments and video views.

Data Source:

Mitchell, J. "Trending YouTube Scraper." GitHub, n.d., https://github.com/mitchelljy/Trending-YouTube-Scraper

About the Dataset:

We used a YouTube trending dataset on the Indian region containing information on videos that went trending on youtube between 1 Dec 2017 and 31 May 2018. It has a total of 37352 entries, all without any missing values. This dataset contains many factors, ranging from the video title and channel name to the published day and how many ratings it has, comprising 16 features.

Methodology:

➔ Importing Libraries and Dataset:

- Imported all the necessary libraries like Numpy, Seaborn, Pandas, Matplotlib, ScikitLearn, Statsmodels, etc
- Imported Dataset.csv through a method called pd.read_csv()

➔ Data Cleaning and preparation

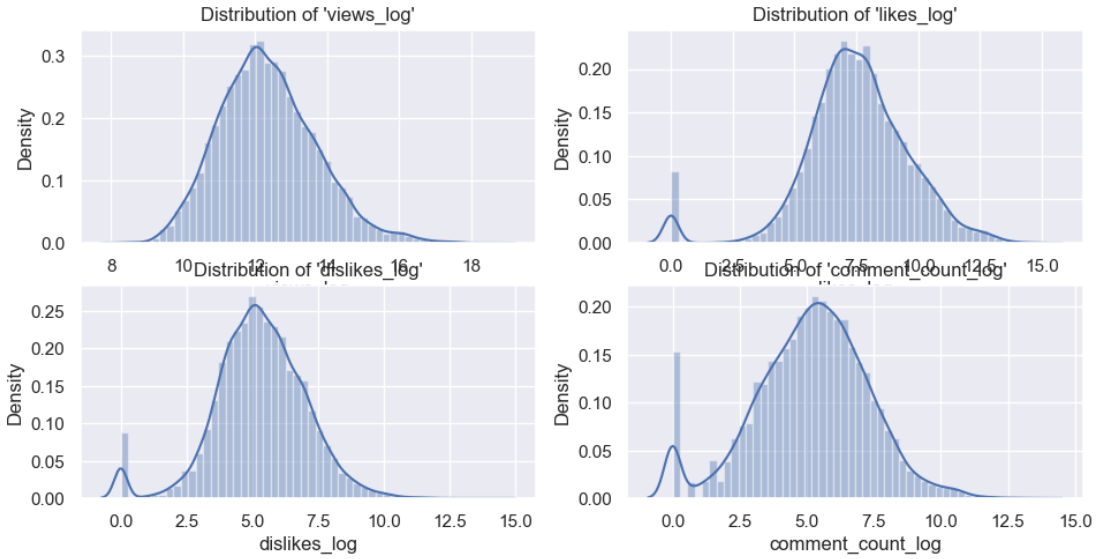
- Upon analysis, we found some rows have some entries as NaN or Null values, so we removed the entire entry.
- A lot of these videos were on trending for multiple days. Because this dataset contains all of the videos trending for each date, these videos are included in the dataset multiple times, so we removed them too.

➔ Statistical and Graphical analysis of the Dataset

- We found that some features like the number of dislikes and likes were heavily skewed and they had a lot more outliers, so we performed the Log transformation of the features
- We did some more Graphical analysis, such as Univariate Analysis, Bivariate Analysis, and Multivariate Analysis.
- We concreted our need to log transform the data from all the analyses we did.

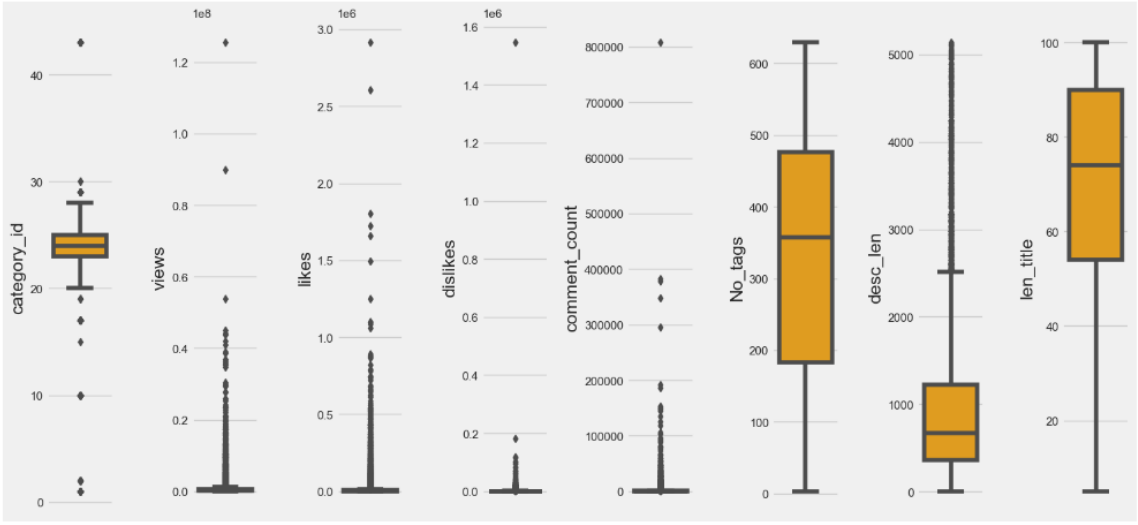
In [49]:

```
# Ploting them again:
fig, plots = plt.subplots(2,2, figsize = (10,5))
plots = plots.flatten()
labels = ['views_log', 'likes_log', 'dislikes_log', 'comment_count_log']
for i in range(len(labels)):
    plots[i] = sns.distplot(data[labels[i]], ax = plots[i])
    plots[i].set_title("Distribution of '{}'".format(labels[i]))
```



In [18]:

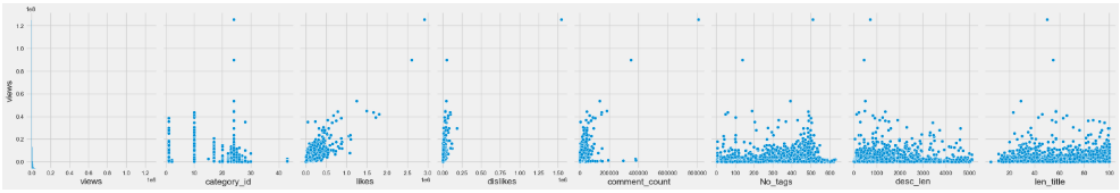
```
# look at the distribution of data with boxplot
features = ['category_id','views','likes','dislikes','comment_count','No_tags','desc_len','len_title']
plt.figure(figsize=(15, 7))
for i in range(0, len(features)):
    plt.subplot(1, 8, i+1)
    sns.boxplot(y=data[features[i]],color='orange',orient='v')
    plt.tight_layout();
#plt.savefig('fig/boxplot.png')
```



Bivariate Analysis

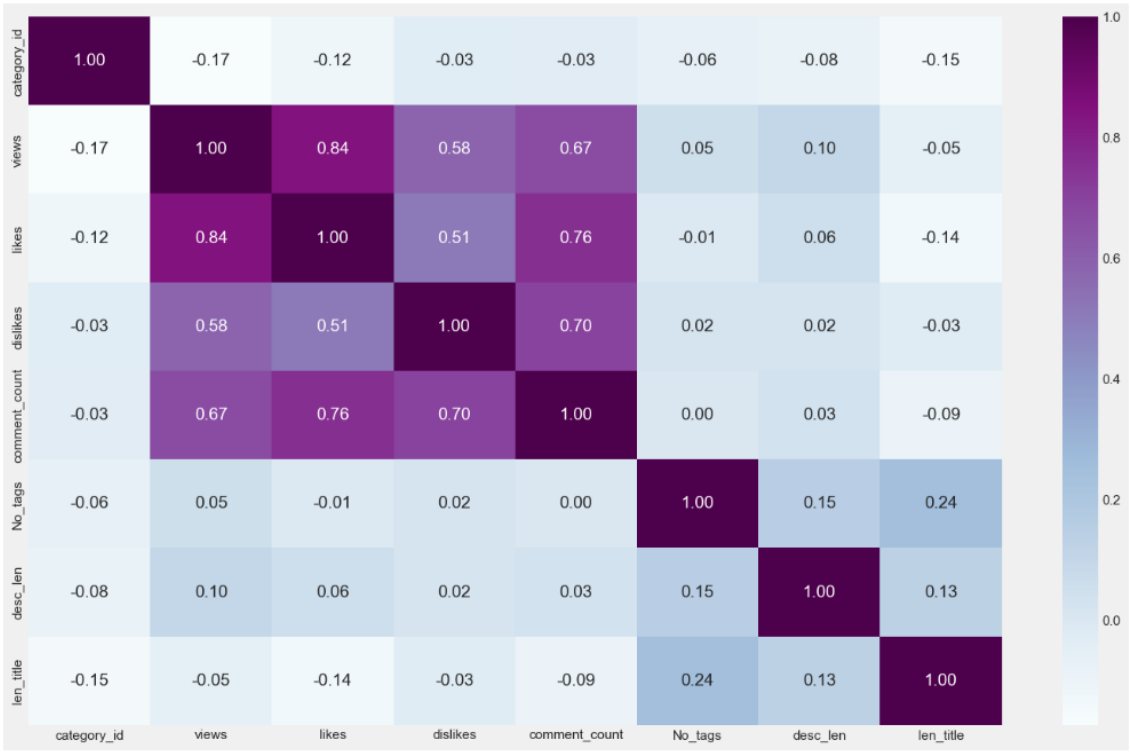
In [20]:
```
#create a pairplot graph from each numeric data
plt.figure(figsize=(10,8))
sns.pairplot(data=data, x_vars=['views','category_id','likes','dislikes','comment_count','No_tags','desc_len','len_title'],
fig.tight_layout();
#plt.savefig('fig/pairplot.png')
```

```
<Figure size 1000x800 with 0 Axes>
```



Multivariate Analysis

In [21]:
```
#create a correlation matrix from each numeric data
features = ['category_id','views','likes','dislikes','comment_count','No_tags','desc_len','len_title']
corr_= data[features].corr()
plt.figure(figsize=(16,10))
sns.heatmap(corr_, annot=True, fmt = ".2f", cmap = "BuPu");
#plt.savefig('fig/heatmap.png');
```



Findings:

- Some Co-relations can be seen between `Views` , `likes` , `dislikes` , `comment_count`

➔ Developing a Hypothesis of the Model

- We are assuming that the following variables can be used to predict the numbers of views of a youtube video
  - The amount of likes on a video (discrete)
  - The amount of dislikes on a video (discrete)
  - The amount of user comments on a video (discrete)
  - The amount of user tags on a video (discrete)
  - Whether the video's rating functionalities were disabled by the creator (categorical: 0, 1)
  - Whether the video's comment functionalities were disabled (categorical: 0, 1)

➔ Data Preparation for the Modeling

- Scaling data
  - Standardizing the independent features in a fixed range. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values higher and consider smaller values as the lower values, regardless of the unit of the values.

```
In [25]:
1 for var in exp_var_num:
2     data_clean["std_"+var]= MinMaxScaler().fit_transform(data_clean[var].values.reshape(len(data_clean), 1))
```

```
In [50]:
1 Data_final.head()
```

Out[50]:

	std_likes_log	std_dislikes_log	std_comment_count_log	std_No_tags	ratings_disabled	comments_disabled	target
12	0.441737	0.279918	0.000000	0.784345	False	True	0.353536
22	0.515826	0.498402	0.482884	0.536741	False	False	0.441346
31	0.509708	0.339930	0.390263	0.383387	False	False	0.403297
37	0.577473	0.440844	0.511304	0.715655	False	False	0.353522
40	0.671623	0.517322	0.545207	0.670927	False	False	0.510718

➔ We used Various models to predict the target variables

- Linear Regression

Linear regression:

In [35]:
```
#Fitting simple linear regression to the Training Set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)
```

Out[35]: LinearRegression()

- Support Vector Regressor Model

Fit Support Vector Regressor Model

In [41]:
```
from sklearn.svm import SVR

svr = SVR(kernel="linear")
svr.fit(xtrain, ytrain)
pred = svr.predict(xtest)
eval_regression(svr, pred, xtrain, ytrain, xtest, ytest)
```

```
MAE: 0.05
RMSE: 0.06
R2 score: 0.77
```

- MLP
  - Default Model

Fit MLP

In [42]:
```
from sklearn.neural_network import MLPRegressor
from sklearn.datasets import make_regression
regr = MLPRegressor(random_state=1, max_iter=500).fit(xtrain, ytrain)
```

In [43]:
```
regr.score(xtest, ytest)
```

Out[43]: 0.7936906714341287

In [44]:
```
nn = MLPRegressor(
    activation='relu',
    hidden_layer_sizes=(10, 100),
    alpha=0.0001,
    solver = "adam",
    random_state=20565,
    early_stopping=False,
)
```

  - Custom Made MLP

In [44]:
```
nn = MLPRegressor(
    activation='relu',
    hidden_layer_sizes=(10, 100),
    alpha=0.0001,
    solver = "adam",
    random_state=20565,
    early_stopping=False,
)
```

In [45]:
```
nn.fit(xtrain, ytrain)
```

Out[45]:
```
MLPRegressor(hidden_layer_sizes=(10, 100), random_state=20565)
```

In [46]:
```
 nn.score(xtest, ytest)
```

Out[46]:
```
0.7980615252842471
```

➔ VIF Analysis

In [47]:
```
# VIF analysis

variables = ['std_likes_log','std_dislikes_log','std_comment_count_log',"ratings_disabled","comments_disabled","std_No_tags"
columns = []
for variable in variables:
    columns.append(X[variable])
baseline = [1 for i in range(Y.count())]
columns.append(baseline)

ck = np.column_stack(columns)

vif = [variance_inflation_factor(ck, i) for i in range(ck.shape[1])]

print("{:>20}: {}".format("Variable", 'VIF'))
for i in range(len(variables)):
    print("{:>20}: {:.1f}".format(variables[i], vif[i]))
```

```
            Variable: VIF
       std_likes_log: 7.6
    std_dislikes_log: 4.4
std_comment_count_log: 6.2
    ratings_disabled: 3.0
   comments_disabled: 2.3
         std_No_tags: 1.0
```

Conclusion:

We tried to predict the number of views of the youtube video by using the significant feature and developed a hypothesis, tested it and rejected it.