Project: YouTube Ad View Prediction

Step 1: Import the different libraries used in data science which will be used for various purposes throughout the project. The libraries included in the project are matplotlib, numpy, sklearn, pandas, seaborn, etc.

Load the dataset using read_csv function from pandas library. The dataset "train.csv" is loaded into the DataFrame "data_train".

Step 2: Perform exploratory data analytics on the loaded dataset to get an overview of the data available. The shape of the data frame is checked first. Now, the data types of each column in the data frame need to be checked using *dtypes*. The data should be all numerical when it is given to a ML model. The categorical or non-numerical data will be converted to the required type during preprocessing.

Using head () to see a sample of the data, the head of the data frame.

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	F
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	D
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	C
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	H
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	D

Step 3: The next step is the preprocessing of the data we have.

It can be observed from the above data frame sample that the data is not of the same type and hence will need to be converted into numerical for input to the ML model.

The following preprocessing is done:

a. The category column contains alphabets and need to be converted into numbers. The alphabets A, B, C, etc are mapped to numbers such as 1, 2, 3, etc respectively.
b. The columns of views, likes, dislikes and comment contain entries such as 'F' which need to be removed from the dataset. Also, the data in these columns in the string format and needs to converted into numeric using the to_numeric () function in pandas.
c. To convert the data in the columns such as vidid, published and duration into numeric or integer type the fit_transform () function of LabelEncoder is used. The label encoder can be imported as – *from sklearn. preprocessing import LabelEncoder*
d. The data in the 'duration' column is the format of hours-minutes-sec and to be able to convert them into seconds we use the time and datetime libraries available.

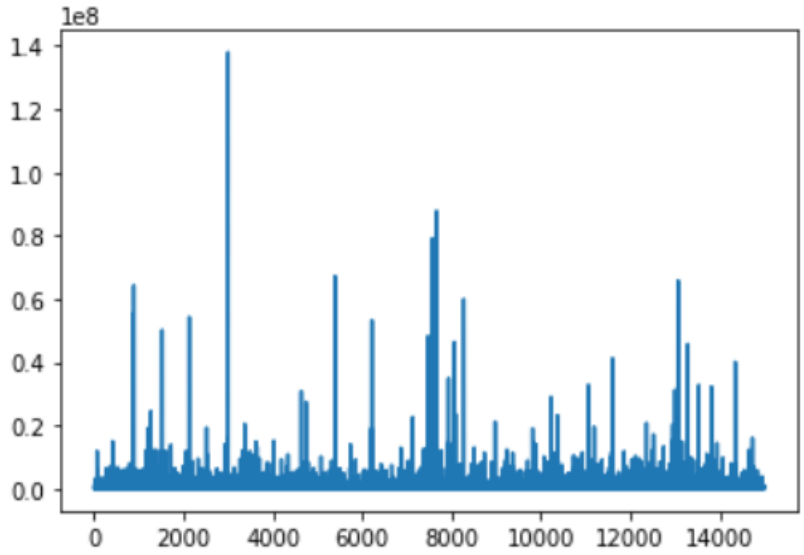
The data after preprocessing:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	5912	40	1031602	8523	363	1095	2168	2925	6
1	2741	2	1707	56	2	6	2185	3040	4
2	8138	1	2023	25	0	2	2094	1863	3
3	9005	6	620860	777	161	153	2119	2546	8
4	122	1	666	1	0	0	2091	1963	4

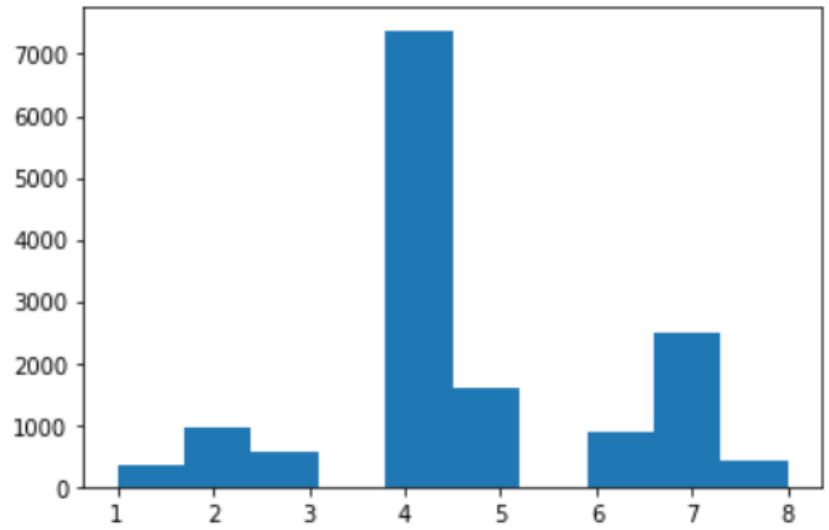
Step 4: Once the data is preprocessed it can now be visualized using the various libraries such as matplotlib for plots and seaborn for heatmaps.

The different column data visualization using matplotlib:

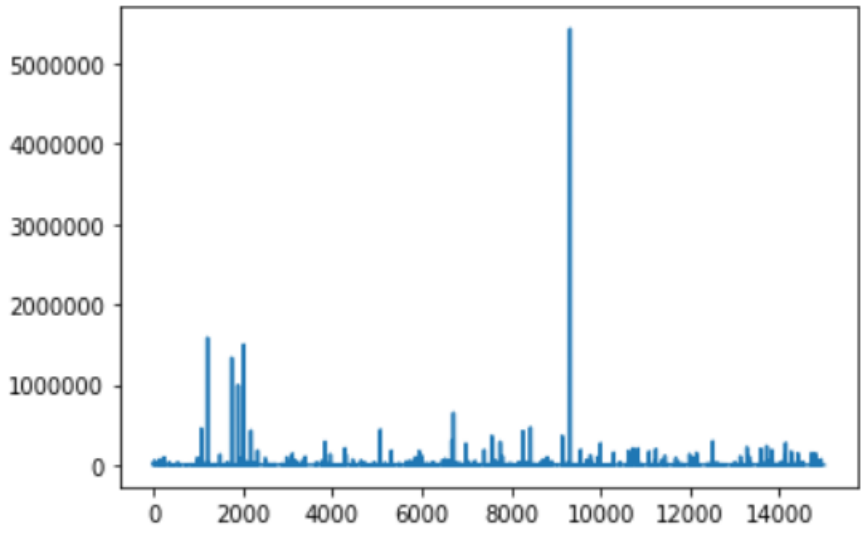
```
#visualizing the processed data
plt.plot(data_train["views"])
plt.show()
```

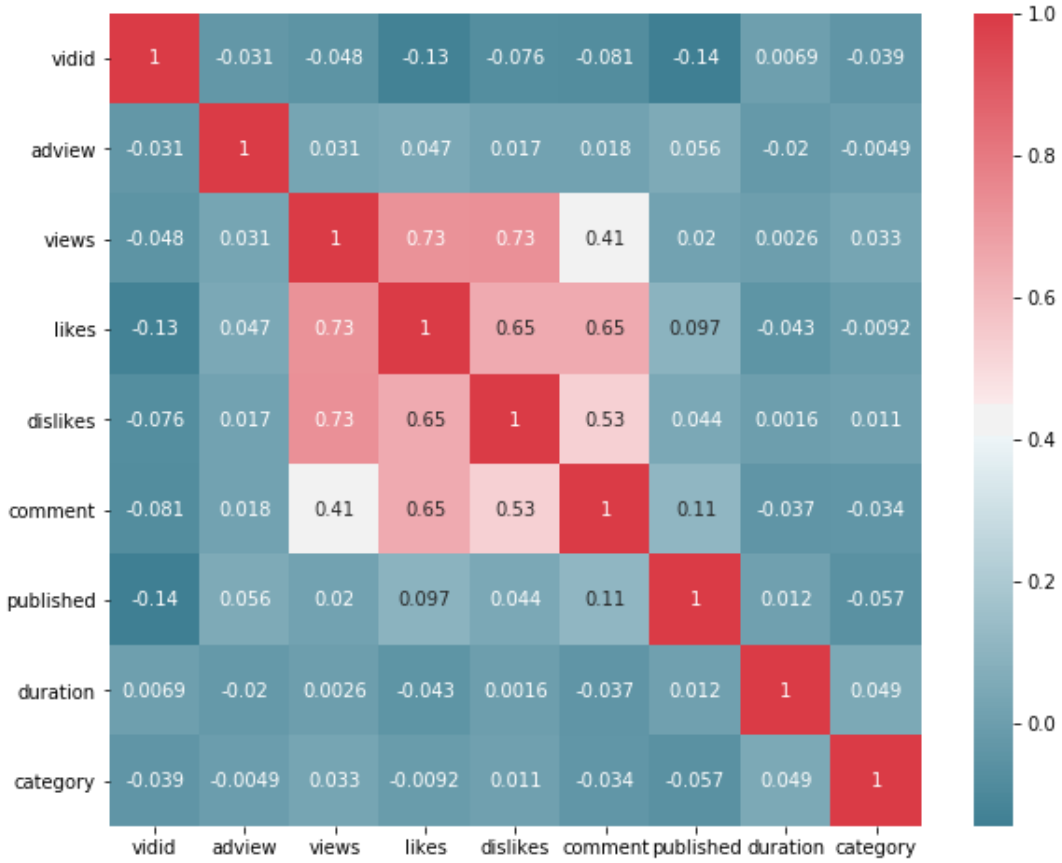


```
plt.hist(data_train["category"])
plt.show()
```




```
plt.plot(data_train["adview"])
plt.show()
```





The heat map shows the correlation between the variables. The closer the value to 1 the more the correlation.

Using visualization, we can get an idea of how exactly the data is and in what range. Looking at the ad views plot, there is an outlier in the data and this needs to be eliminated or else it will affect the model prediction.

Step 5: The data is now ready to be given as input to the machine learning model.

Before we select the machine learning model, the input variables and the target variables should be analyzed. In this project, the 'adviews' column is the target variable. the target variable is the column that we want our model to predict.

The column adviews is being copied in another data frame named '*data_target'* for our reference and to cross check on our model predictions.

The columns 'adviews' and 'vidid' are being dropped from the data frame data_train. The updated data frame and the one to be given as input to the model is –

	views	likes	dislikes	comment	published	duration	category
0	1031602	8523	363	1095	2168	457	6
1	1707	56	2	6	2185	570	4
2	2023	25	0	2	2094	136	3
3	620860	777	161	153	2119	262	8
4	666	1	0	0	2091	31	4

Step 6: The data is now split into training dataset and test dataset. The training dataset is used to train our machine learning model and make it learn. The testing data is unseen data which will be used to see how well was our model trained and what is predicts using unseen data.

This split is done using the *train_test_split ()* function which can be imported as – from *sklearn. model_selection import train_test_split.* The split of data in this project is in the ratio 80:20 and random state is assigned to 42.

```
#training and testing data split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_train, data_target, test_size=0.2, random_state=42 )
```

X_train, y_train contains the input and target training data and X_test, y_test is the data used to test the model on the unseen data.

Step 7: The Data is now split into two parts training dataset and testing dataset. It can be observed that in the input data the values vary significantly.

The technique of normalizing the data is used to get all the values in a particular range so that the values that are way ahead in the range of the data do not impact the model training. This may effect in model overfitting if not normalized.

In the project the MinMaxScaler is used to normalize the data.

```
#normalizing the input data to reduce the data items to the range 0 to 1
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

Step 8: The different regression models are implemented and the mean absolute error and the mean squared error for each model is calculated to help choose the best model for our project.

The different models implemented are Linear regression, Decision tree regressor, random forest, support vector regressor. The MAE and MSE for each model are computed. The model having the least error is selected.

The functions fit () and predict () and used to train the model and to make prediction respectively. The fit () function takes X_train and y_train as arguments while the predict () function takes X_test are the function argument.

```
#creating a function to calculate errors for each each model
from sklearn import metrics
def get_error(X_test, y_test, model_name):
    prediction = model_name.predict(X_test)
    print("Mean Absolute Error:", metrics.mean_absolute_error(y_test, prediction))
    print("Mean Squared Error:", metrics.mean_squared_error(y_test, prediction))
```

Step 9: The best model is finalized based on the error scores. This model is now saved. The libraries pickle or joblib are used for the same. In this project the joblib library is used to save the model and then load the model to test on unseen data.

```
#saving and then loading the SVR model to file using joblib
import joblib
joblib_file = "joblib_DT_model.pkl"
joblib.dump(DT , joblib_file)
DT_model = joblib.load(joblib_file)
DT_model
```

This model is then loaded to make future predictions on any unseen data.

Step 10: The make the predictions for the unseen data, the csv file named *'test.csv'* is loaded into a data frame *'data_test'*.

This data frame consists of unseen data which needs to be preprocessed as done earlier with the training data.

The *data_test* after preprocessing is –

	views	likes	dislikes	comment	published	duration	category
0	440238	6153	218	1377	2053	449	2
1	1040132	8171	340	1047	1825	389	6
2	28534	31	11	1	1009	2274	4
3	1316715	2284	250	274	116	595	7
4	1893173	2519	225	116	1892	188	2

This data is now given as an input to the machine learning regression model that we finalized earlier. The model is loaded and takes the input as *data_test*. An additional column of adviews is added to the data frame. The predictions given as an output by the model is then stored in the adviews column. The *data_test* along with adviews is finally converted into a csv file using to_csv () function and saved as '*Predictions_Submission.csv*'