ENGINNERING FACULTY - COMPUTER ENGINEERING DEPARTMENT

MACHINE LEARNING
2022-2023 SPRING
FINAL PROJECT REPORT

INSTRUCTOR : Assoc. Prof. Dr. Ahmet Çağdaş SEÇKİN

GROUP MEMBERS	
STUDENT NAME	**STUDENT ID**
Esmanur DELİ	**191805056**
Kübra UÇAR	**191805067**
Rabia YILDIRIM	**191805043**
Ebrar DEMİR	**191805105**

PROJECT SUBJECT:

Applying machine learning algorithms on the dataset we have prepared by searching for different key values on twitter.

1 INSTRUCTIONS AND TABLE OF CONTENTS

1.1 TABLE OF CONTENTS

2 COMMON DATASET PROJECT

2.1 DATASET AND PREPROCESSING

- **Data Source:** Doç.Dr.Ahmet Çağdaş Seçkin

- **Data Description:**

  These are our columns.

```
X (Numeric)          float64
Y (Numeric)          float64
Floor (Categoric)      int64
F_dB_min               int64
F_dB_max               int64
                      ...
CH_98                  int64
dB_98                  int64
MAC_ID_99              int64
CH_99                  int64
dB_99                  int64
Length: 315, dtype: object
```

Null count: 0
Number of columns and rows:

```
(1985, 315)
```

'X (Numeric)', 'Y (Numerical)', 'Floor (Categorical)' these are the columns we will guess. The remaining columns are our attributes.

- **Data Split:**

```
X = data.drop(['X (Numeric)', 'Y (Numeric)', 'Floor (Categoric)'], axis=1)
X_numeric = data['X (Numeric)']
Y_numeric = data['Y (Numeric)']
floor = data['Floor (Categoric)']
```

Distinction for this X numeric value

```
X_train, X_test, y_train, y_test = train_test_split(X,X_numeric, test_size=0.3, random_state=100)
```

Distinction for this Y numeric value

```
X_train, X_test, y_train, y_test = train_test_split(X,Y_numeric, test_size=0.3, random_state=100)
```

Distinction for this Floor value

```
#Floor
X_train, X_test, y_train, y_test = train_test_split(X,floor, test_size=0.3, random_state=100)
```

- **Data Exploration:**

First two columns:

Index	X (Numeric)	Y (Numeric)	loor (Categoric	F_dB_min	F_dB_max	F_dB_mean	F_dB_std	dB_MAC_ID_n	dB_MAC_ID_m	F_dB_CH_min	F_dB_CH_max	F_nofCH
0	0	0	0	-40	-82	-69.1579	12.0243	7	11	1	6	3
1	0	0	0	-40	-82	-69.1579	12.0243	7	11	1	6	3

Here we can see the numerical values of the columns:

```
       X (Numeric)  Y (Numeric)  ...        CH_99        dB_99
count  1985.000000  1985.000000  ...  1985.000000  1985.000000
mean      0.781864    12.070025  ...     0.024181    -0.303275
std       9.360245    12.381293  ...     0.380232     4.769137
min     -13.200000    -6.000000  ...     0.000000   -76.000000
25%      -8.400000     0.000000  ...     0.000000     0.000000
50%       1.200000     9.600000  ...     0.000000     0.000000
75%       9.600000    22.800000  ...     0.000000     0.000000
max      18.000000    34.800000  ...     6.000000     0.000000

[8 rows x 315 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1985 entries, 0 to 1984
Columns: 315 entries, X (Numeric) to dB_99
dtypes: float64(4), int64(311)
memory usage: 4.8 MB
```

- **Data Preprocessing:**

Since there is a space at the beginning of the columns, I first deleted these spaces.

```
#Dataset
data.columns = data.columns.str.strip()
```

There are no null values.
There are no duplicate values.

We brought the features closer together with the scale operation.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = scaler.fit_transform(X)
```

- **Data Usage:**

X and Y positions will be calculated with random forest, linear and knn and the best values will be selected according to r2 scores.

Floor classification will be trained with randomforest, adaboost and knn algorithms, and the best model will be found according to the accuracy value.

In this way, the best position where the device can be placed will be determined.

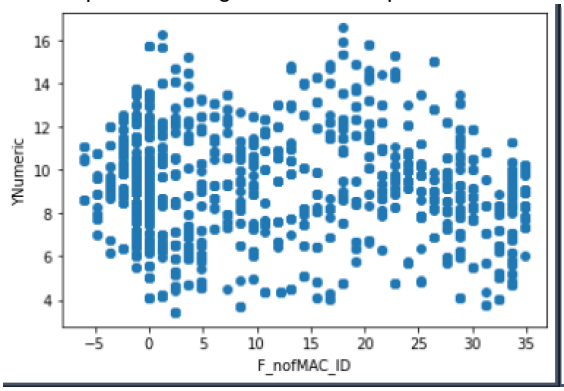
- **Feature engineering:**

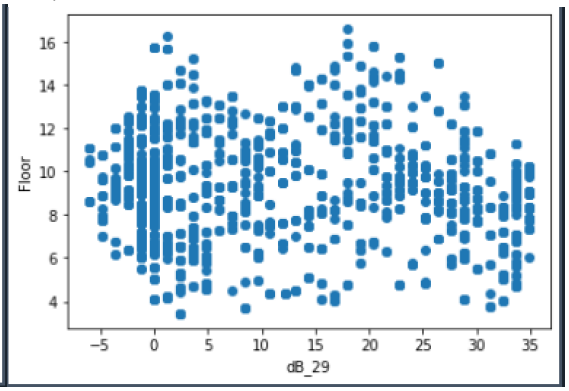
Since there are 315 columns, it is not possible to show the effects of all features, so it would be more accurate to give the weights of 10 features that affect the best model, the random forest.

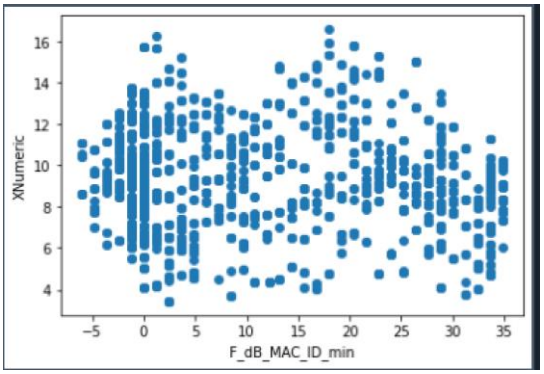
Feature	Importance
F_dB_MAC_ID_min	0.0398157
CH_18	0.0217274
dB_0	0.0191818
F_dB_mean	0.0187943
CH_35	0.0174314
dB_29	0.0167063
dB_9	0.0154442
dB_10	0.0143904
F_nofMAC_ID	0.0140988
dB_3	0.0135967
MAC_ID_80	0.0135235
CH_80	0.0134865
MAC_ID_0	0.0134274
dB_17	0.0126867

- **Dataset visualization:**

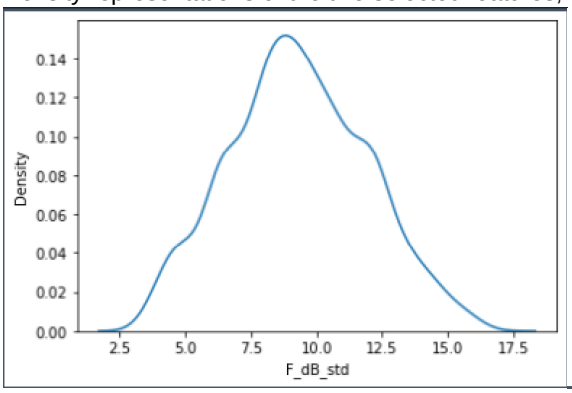
Scatter plots showing the relationship of the two features;

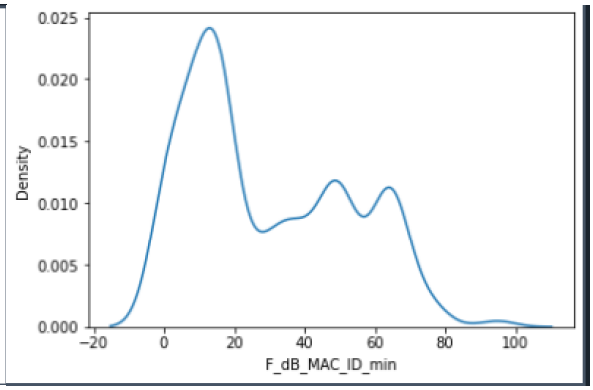




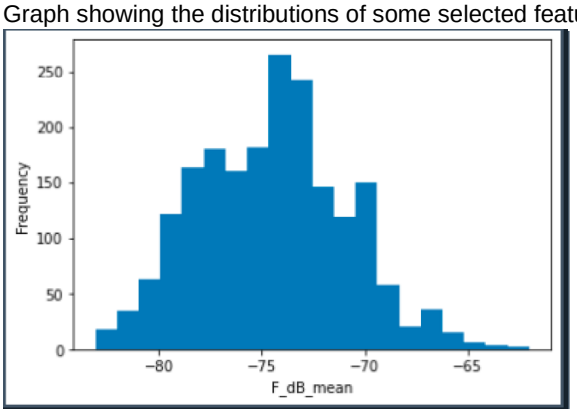


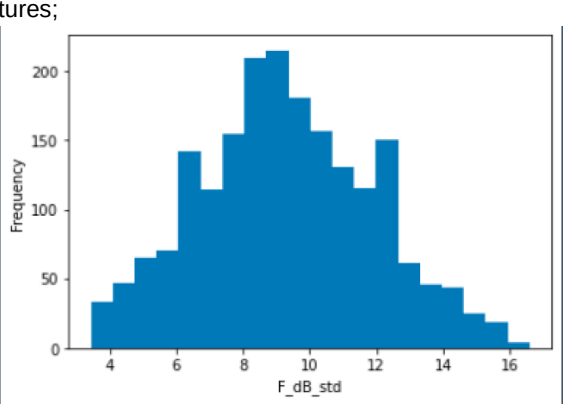
Density representations of the two selected features;

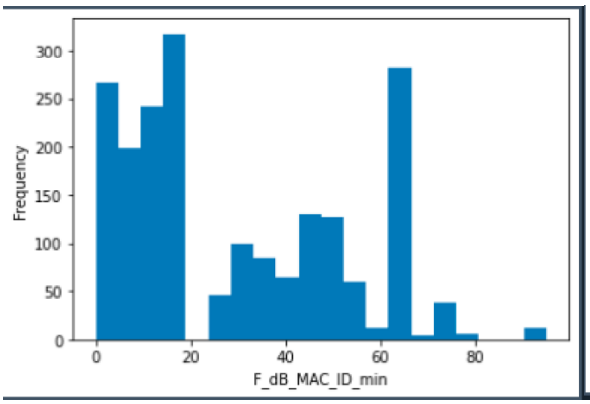


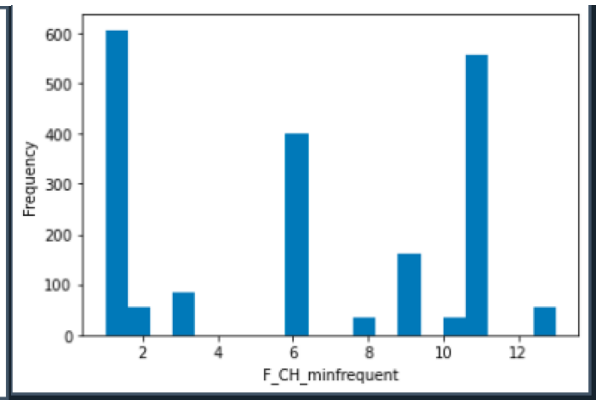


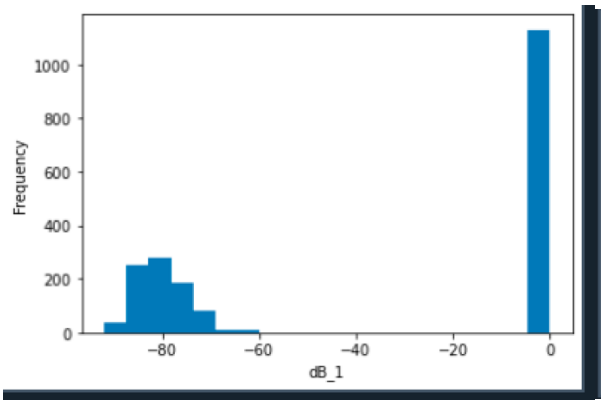
Graph showing the distributions of some selected features;

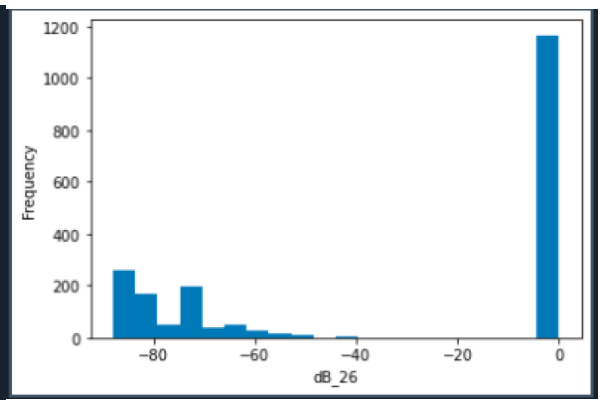












2.2 COMMON DATASET RESULTS AND MODEL SELECTION

- Use multiple algorithms to get the best performance.

Three different algorithms were used for regression; RandomForest, Linear Regression and KNN

For classification, 3 different algorithms were used, these are random forest, adabood and knn algorithms.

The values of these algorithms according to 3 different labels are given below. According to R2 values and mse values

For the value of X

```
Test Mean Squared Error Random Forest X_numeric: 0.16516120134228104
Test R-squared Score Random Forest X_numeric: 0.9980536674670796
Train Mean Squared Error Random Forest X_numeric: 0.03226979409647192
Train R-squared Score Random Forest X_numeric: 0.9996363131777614
Mean Squared Error Linear X_numeric: 2.311246763065074
R-squared Linear X_numeric: 0.9727632474818462
Train Mean Squared Error RLinear X_numeric: 0.03226979409647192
Train R-squared Score Linear X_numeric: 0.9996363131777614
Mean Squared Error knn: X_numeric 2.416375167785235
R-squared Score knn: X_numeric 0.9715243679352096
Train Mean Squared Error KNN X_numeric: 0.03226979409647192
Train R-squared Score KNN X_numeric: 0.9996363131777614
```

For the value of Y

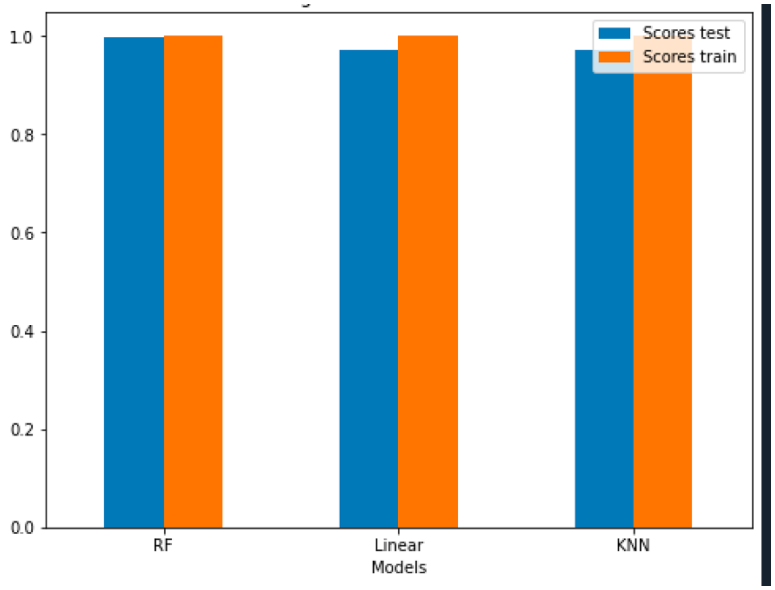
```
Test Mean Squared Error Random Forest Y_numeric: 0.9969182283287359
Test R-squared Score Random Forest Y_numeric: 0.9969182283287359
Train Mean Squared Error Random Forest Y_numeric: 0.08478065658747322
Train R-squared Score Random Forest Y_numeric: 0.9994344798772857
Mean Squared Error Linear X_numeric: 5.3165079485877245
R-squared Linear Y_numeric: 0.9669604809185443
Train Mean Squared Error Linear Y_numeric: 0.08478065658747322
Train R-squared Score Linear Y_numeric: 0.9994344798772857
Test Mean Squared Error knn: Y_numeric 4.872418791946308
Test R-squared Score knn: Y_numeric 0.9697202797012432
Train Mean Squared Error knn _numeric: 0.08478065658747322
Train R-squared Score knn Y_numeric: 0.9994344798772857
```

For the value of Floor

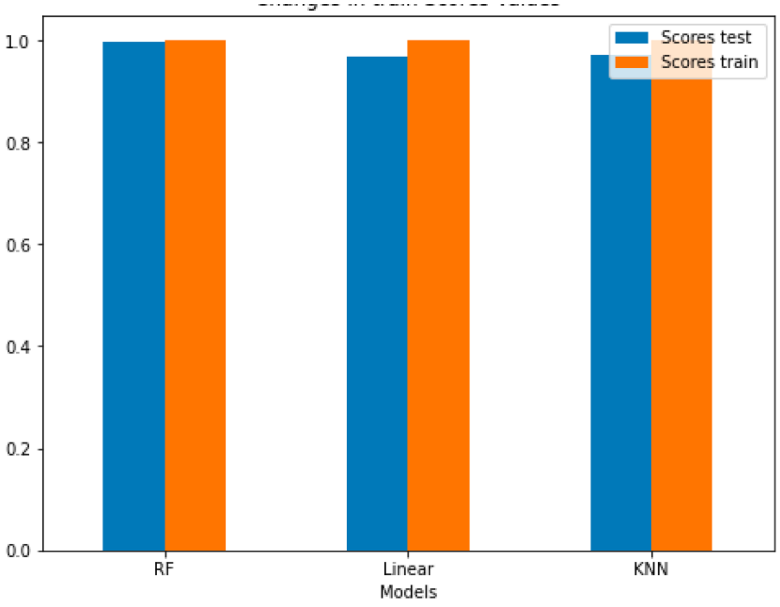
```
Accuracy ada test: 0.5553691275167785
Accuracy ada train: 0.5284377249820015
Accuracy knn test: 0.9412751677852349
Accuracy knn train: 0.9834413246940245
Accuracy rf test: 1.0
Accuracy rf train: 0.9834413246940245
```

- Compare training results. Create a comparison table. And Compare test results. Create a comparison table.

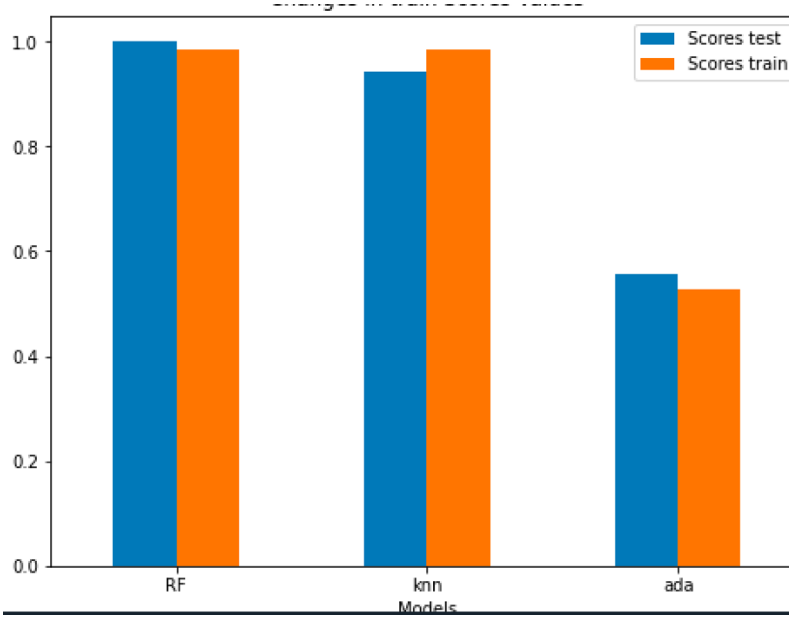
Visualization of train and test results for X value



Visualization of train and test results for Y value



Visualization of train and test results for Floor value



- Select the best training algorithm.
  For the X value that gives the best r2 value in the regression according to the test results: Linear regression with 0.9980
  For Y value: knn algorithm with 0.9969

  We look at the one with the highest accuracy value for the Classification.
  For floor value: It is a random forest model with 1.0.

  ACHIEVEMENT VALUE = 0.9980 * 0.9969 * 1
  = 0.9949062

- Try to reduce test and training time.
  Training time + test time =

```
Time: 8.428308963775635
```

- Try to improve performance metric.
  When we made a Standardscaler, it was seen that the classification model gave better results.
  The incremental results are below.

```
Accuracy ada test: 1.0
Accuracy ada train: 1.0
Accuracy knn test: 0.9412751677852349
Accuracy knn train: 0.9834413246940245
Accuracy rf test: 1.0
Accuracy rf train: 0.9834413246940245
```

3 ORIGINAL DATASET REGRESSION PROJECT

3.1 DATASET AND PREPROCESSING

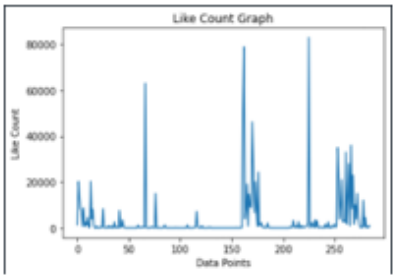
- **Data Source:** www.twitter.com
- **Data Description:**

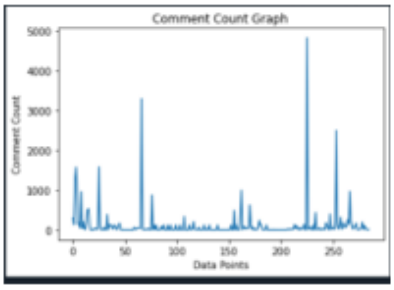
```
RangeIndex: 284 entries, 0 to 283
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Id                  284 non-null    int64
 1   Like Count          284 non-null    int64
 2   Comment Count       284 non-null    int64
 3   retweet Count       284 non-null    int64
 4   View count          284 non-null    int64
 5   countOfWords        284 non-null    int64
 6   countOfPositive     284 non-null    int64
 7   countOfNegative     284 non-null    int64
 8   Art                 284 non-null    int64
 9   Health              284 non-null    int64
 10  Politics            284 non-null    int64
 11  Science             284 non-null    int64
 12  Sport               284 non-null    int64
 13  C_Text__mysteries   284 non-null    float64
 14  C_Text__replying    284 non-null    float64
 15  T_Text__replying    284 non-null    float64
 16  T_Text__manchester  284 non-null    float64
 17  C_Text__manchester  284 non-null    float64
 18  T_Text__believe     284 non-null    float64
 19  T_Text__mark        284 non-null    float64
dtypes: float64(7), int64(13)
```

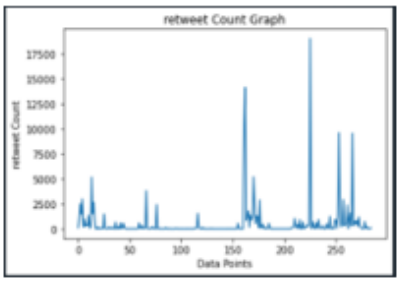
A dataset with 284 rows and 19 columns. The final state of this data after preprocessing
Like Count = Shows how many likes the tweet received
Comment Count = Shows how many comments the tweet received
Retweet Count = Shows how many retweet the tweet received
View Count = Shows the number of times the tweet was viewed
countOfWords = Shows how many words the tweet consists of
countOfPositive = Returns the number of positive words in the tweet. This number was determined according to the words in the data set we prepared ourselves.
countOfNegative = Returns the number of negative words in the tweet. This number was determined according to the words in the data set we prepared ourselves.
Art,Health,Politics,Science,Sport = These columns show which sector the tweet is in, for example, if the tweet is in the Arts sector, it would say 1 if not 0.
C_Text__mysteries,C_Text__replying,T_Text__replying,T_Text__manchester,C_Text__manchester,T_Text__believe,T_Text__mark = These columns are the columns that make the biggest contribution to the data as a result of the Tf-idf and count vectorization processes of the tweet content.

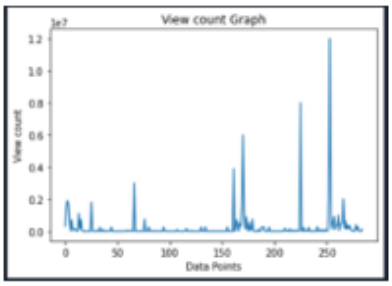
- **ML Problem definition:** According to some of the features we get from twitter (retweets, views, number of comments..... etc.) estimating the number of likes of a tweet.
- **Data Split:** 80% of the dataset is reserved for train and 20% for testing.

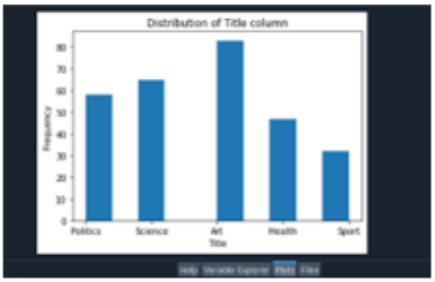
- **Data Exploration:** Some Columns Hist graphics shown as below, you can see the distribution on datas











Data Preprocessing:

First of all, we read the unprocessed dataset and then applied this steps:

1 -

```
dataset.drop_duplicates(inplace=True)
```

2 - Instead of the K and M values in the comment, retweet and view columns, we convert those columns into numerical data by typing their numerical equivalents.

```
like_column = 'Like Count'
comment_column = 'Comment Count'
retweet_column = 'retweet Count'
view_column = 'View count'

dataset[like_column]

def convert_k_m(column_name):
    for i in range(len(dataset)):
        deger = str(dataset.at[i, column_name])
        if 'K' in deger:
            deger = deger.replace('K', '')
            deger = float(deger) * 1000
            dataset.at[i, column_name] = int(deger)
        elif  'M' in deger:
            deger = deger.replace('M', '')
            deger = float(deger) * 1000000
            dataset.at[i, column_name] = int(deger)

convert_k_m(like_column)
convert_k_m(comment_column)
convert_k_m(retweet_column)
convert_k_m(view_column)
```

3 - Here, we fill in the missing values with the average of the column using the mean() function. That is, the missing values are replaced by the average of the other values in that column.

```
dataset[comment_column].fillna(dataset[comment_column].mean(), inplace=True)
dataset[comment_column] = dataset[comment_column].astype(int)
dataset[like_column].fillna(dataset[like_column].mean(), inplace=True)
dataset[like_column] = dataset[like_column].astype(int)
dataset[view_column].fillna(dataset[view_column].mean(), inplace=True)
dataset[view_column] = dataset[view_column].astype(int)
dataset[retweet_column].fillna(dataset[retweet_column].mean(), inplace=True)
dataset[retweet_column] = dataset[retweet_column].astype(int)
```

4 - We start to infer meaning in the text column. These text-mining steps:

We calculate the number of words in the text content and save it to the countOfWords column and we find the number of positive and negative words in the text content, based on the data set we prepared ourselves last term. And we save them in countOfNegative and countOfPositive columns.

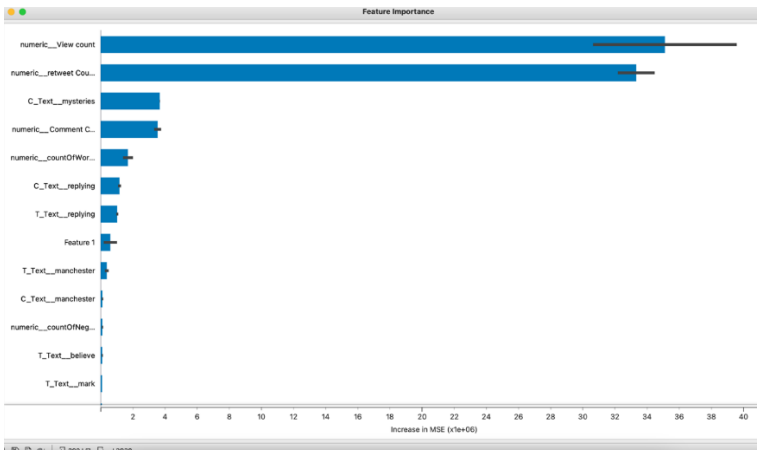
```
text_column = dataset["Text"]
#kelime sayısını ayırma
def countOfWord(text):
    liste = []
    for x in range(0,len(text)):
        liste.append(len(text[x].split()))
    return liste

countOfText = countOfWord(text_column)
dataset["countOfWords"] = countOfText

data_positive = pd.read_csv("PositiveWordsEng.csv")
data_negative = pd.read_csv("NegativeWordsEng.csv")
data_positive = data_positive["PositiveWords"]
data_negative = data_negative["NegativeWords"]

def search_tw(data_words,data_tw):
    liste = []
    for tweet in data_tw:
        count = 0
        tweet = str(tweet).lower().split()
        for word in data_words:
            if word in tweet:
                count+=1
        liste.append(count)
    return liste

countOfPositive = search_tw(data_positive,text_column)
countOfNegative = search_tw(data_negative,text_column)


dataset["countOfPositive"] = countOfPositive

dataset["countOfNegative"] = countOfNegative

dataset.to_csv("dataset_c.csv",index=False)
```

5 - We perform text-mining by applying TfIdf and count vectorixation operations to the text column. Since there are about 4 thousand columns here, we make feature importance on Orange and add the ones that give the best results according to r2 and MSE values to the data set.

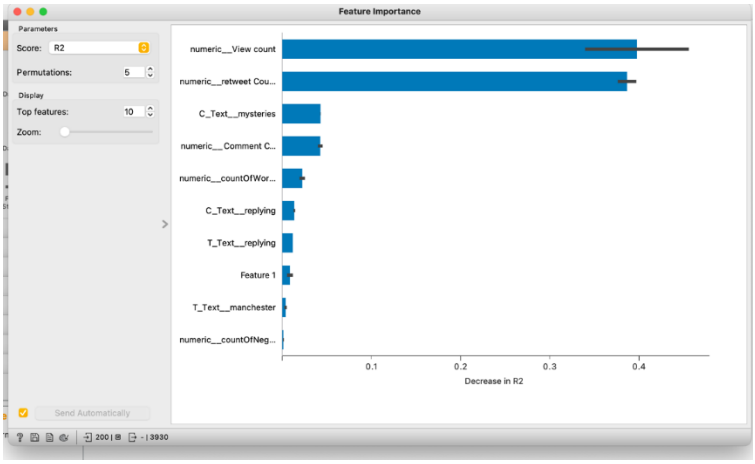
```
# Preprocess text data
Tfidf_transformer = TfidfVectorizer(stop_words=my_stop_words)
count_transformer = CountVectorizer(stop_words=my_stop_words)
scaler = StandardScaler()

# Combine text and numeric features
preprocessor = ColumnTransformer(transformers=[
    ("T_Text", Tfidf_transformer, "Text"),

    ('C_Text', count_transformer, 'Text'),
    ('numeric', 'passthrough', coulmn_names)
])
# Fit and transform the preprocessor to the data
X_preprocessed = preprocessor.fit_transform(X)
```

The values in Orange;

MSE:



R2 score:

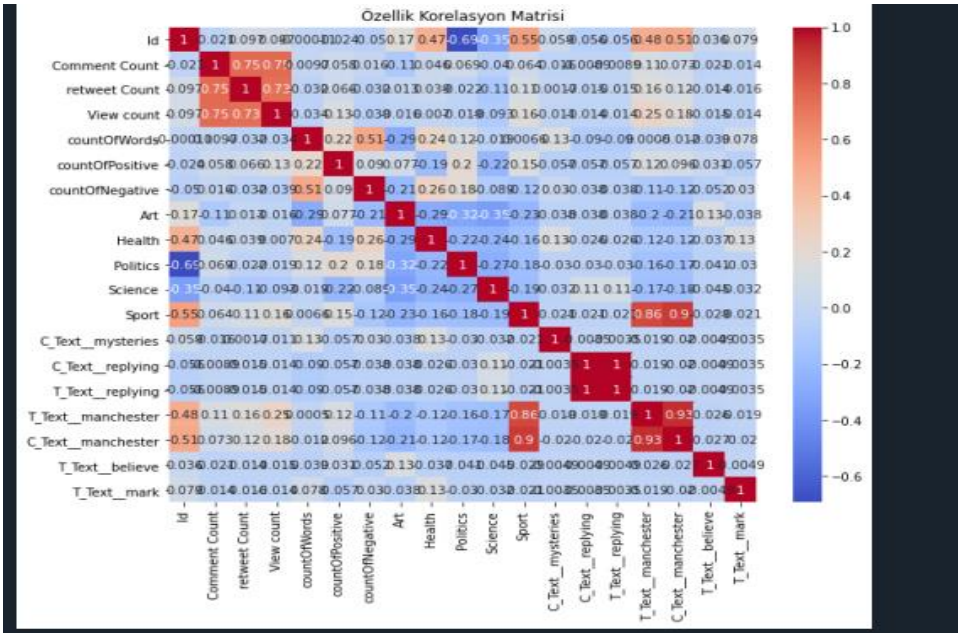


6-The title column was converted to a numeric value by one hot encoding.

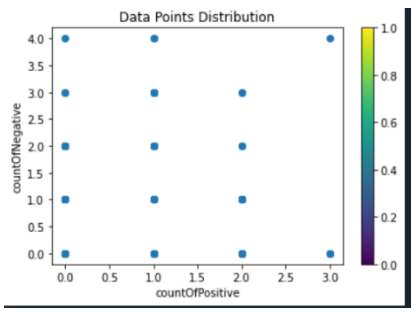
```
#for regression

one_hot_df = pd.get_dummies(dataset['Title'])
dataset['ID'] = range(1, len(dataset) + 1)
one_hot_df['ID'] = range(1, len(one_hot_df) + 1)


one_hot_df[one_hot_df == True] = "1"
one_hot_df[one_hot_df == False] = "0"

merged_df = pd.merge(dataset, one_hot_df, on='ID')

merged_df = merged_df.drop('ID',axis=1)
merged_df = merged_df.drop('Title',axis=1)
merged_df.to_csv("dataset_reg.csv",index=False)
```

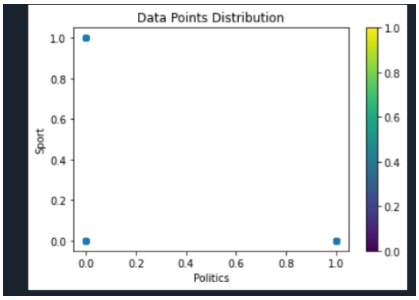
- **Data Usage:** Machine learning algorithms such as AdaBoost ,Random Forest, Lineer Regression will be used. According to these algorithms, Tweet like count will be predicted.

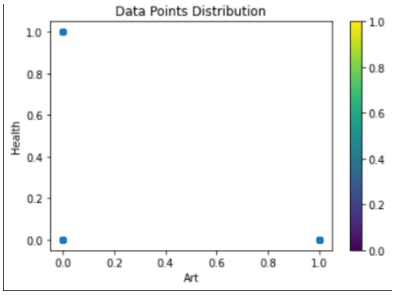
- **Feature engineering:** Since we use our own dataset, the correlation value of most of our features is close to 0, but the correlation value between tweet likes, comments, retweets, and views is close to 1.

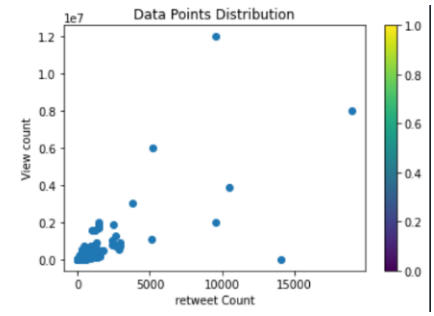


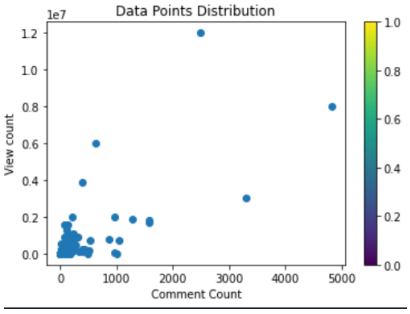
- **Dataset visualization:** We did the data distribution analysis for some features.











3.2 REGRESSION STEPS

1.Upload the dataset and import the necessary libraries:

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

2. Load the data:

```
regression_df = pd.read_csv('data_preprocessing/reg_dataset.csv')
```

3.Determine the dependent variable and the independent variables:

```
X = regression_df.drop(['Like Count'], axis=1)
y = regression_df['Like Count']
```

4. Divide the data into training and test sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=100)
```

5. Create and train the Random Forest Regression model:

```
# Random Forest Regresyon modelini oluşturun
regressor = RandomForestRegressor(n_estimators=100, random_state=0)

# Modeli eğitin
regressor.fit(X_train, y_train)
```

6. Make predictions with the Random Forest Regression model:

```
y_pred_rf = regressor.predict(X_test)
```

7. Evaluate the performance of the Random Forest Regression model:

```
# Test verileri üzerinde tahmin yapın
y_pred = regressor.predict(X_test)

# Model performansını değerlendirin
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R-squared Score:', r2)
```

8. Create and train the AdaBoost Regression model:

```
# AdaBoost regresyon modelini oluşturma ve modeli eğitin
ada_regressor = AdaBoostRegressor(n_estimators=100, learning_rate=1.0, random_state=42)
ada_regressor.fit(X_train, y_train)
```

9. Make predictions with the AdaBoost Regression model:????????????

```
y_pred_ada = ada_regressor.predict(X_test)
```

10. Evaluate the performance of the AdaBoost Regression model:

```
# Eğitim ve test verileri üzerinde tahmin yapma
y_train_pred = ada_regressor.predict(X_train)
y_test_pred = ada_regressor.predict(X_test)

# Eğitim ve test MSE değerini hesaplama
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)

# Eğitim ve test R-kare değerini hesaplama
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

print("Eğitim MSE:", train_mse)
print("Test MSE:", test_mse)
print("Eğitim R-kare:", train_r2)
print("Test R-kare:", test_r2)
```

11. Create and train a Linear Regression model:

```
# Linear regresyon modelini oluşturma ve eğitme
model = LinearRegression()
model.fit(X_train, y_train)
```

12. Make predictions with a Linear Regression model:

```
y_pred_linear = model.predict(X_test)
```

13. Print the coefficients and breakpoint of the trained linear regression model:

```
# Linear regresyon modelini oluşturma ve eğitme
model = LinearRegression()
model.fit(X_train, y_train)

# Eğitilmiş modelin katsayıları ve kesme noktası
print("Katsayılar:", model.coef_)
print("Kesme Noktası:", model.intercept_)

# Tahmin yapma
y_pred = model.predict(X_test)
```

In general, regression loads the dataset, separates the training and test datasets, creates and trains three different regression models (Random Forest, AdaBoost, Lineer Regression).According to results, Lineer Regression is the best model.You can see below:

```
Random Forest:
Train MAE: 489.34378854625555
Train R2: 0.9731656597601084
Test MAE: 2776.3756140350883
Test R2: 0.6369724628323925

AdaBoost:
Train MAE: 1390.5422084267598
Train R2: 0.9642238325838217
Test MAE: 3541.6712232496757
Test R2: 0.6333281720943157

Linear Regression:
Train MAE: 2083.362744368039
Train R2: 0.7622865842301308
Test MAE: 3195.5638434857306
Test R2: 0.8511556012001812
```

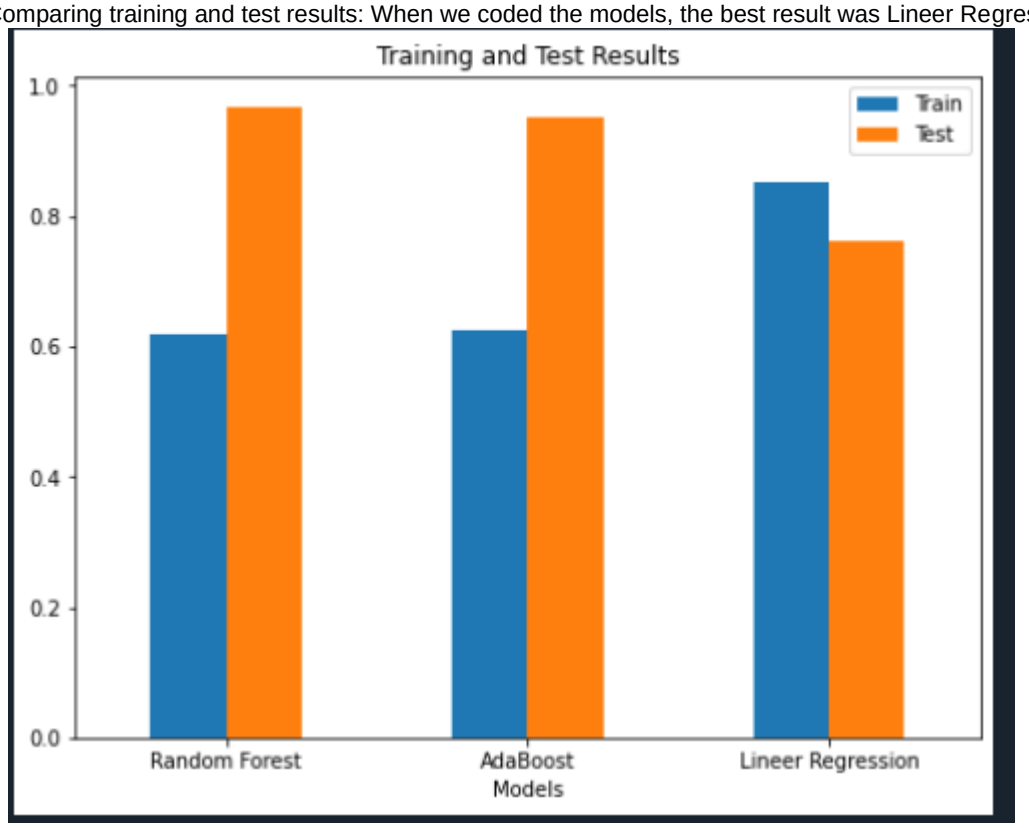
3.3 REGRESSION RESULTS AND MODEL SELECTION

Three different algorithms were used for the regression: Random Forest, Linear Regression, AdaBoast. The best model is Lineer Regression.

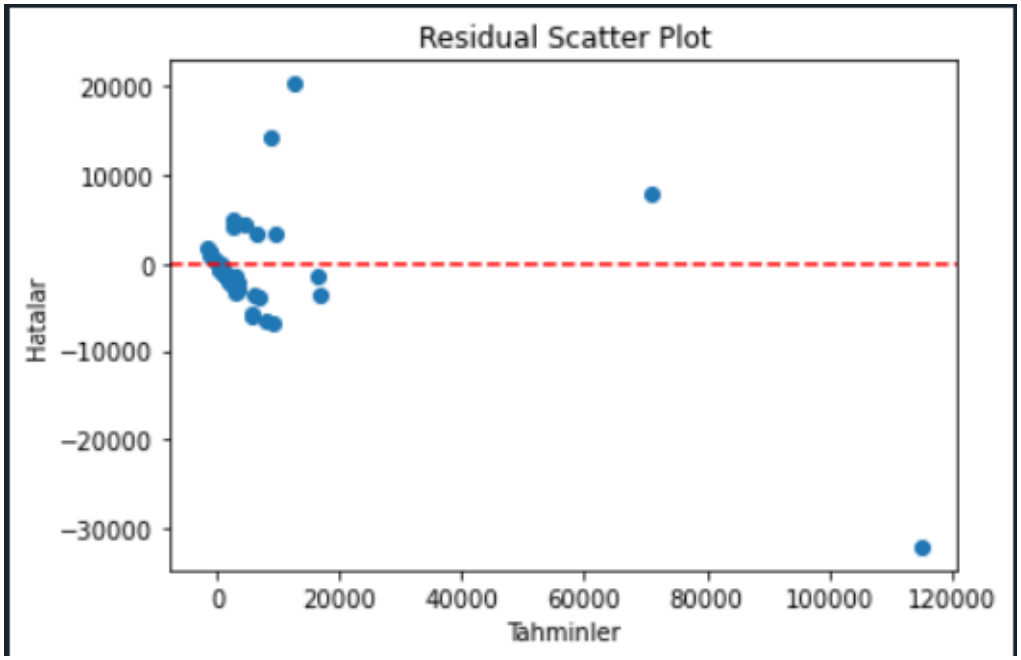
We have tried different models through orange and you can see the results of these models below:

Model	MSE	RMSE	MAE	R2
kNN	40058021.128	6329.141	1618.238	0.604
SVM	110601147.006	10516.708	3210.336	-0.092
Random Forest	16170489.543	4021.255	969.290	0.840
Linear Regression	16412010.769	4051.174	2075.821	0.838
AdaBoost	27513.772	165.873	86.579	1.000

Comparing training and test results: When we coded the models, the best result was Lineer Regression.



The best model was linear regression, and you can see the residual scatter plot of it below:



3.4 REGRESSION HYPERPARAMTER OPTIMIZATION RESULTS

- Try to reduce test and training time.
  we tried to improve the linear regression model by using different hyperparameters and improved the mae value, albeit very slightly

```
Linear Regression:
Train MAE: 2083.362744368039
Train R2: 0.7622865842301308
Test MAE: 3195.5638434857306
Test R2: 0.8511556012001812
Optimistic MAE: 3195.563843486241
Optimistic R2: 0.8511556012001312
```

Train and Test Time: We have not a big dataset so the time is good. But for huge dataset, we can do for reducing the time these:

1. Reduce the size of the dataset by selecting a smaller subset of data.
2. Perform feature selection or feature extraction to eliminate irrelevant or redundant features.
3. Utilize parallel processing techniques such as multi-threading or multiprocessing to train and test the model simultaneously.
4. Apply techniques like random under-sampling or random over-sampling to balance the dataset and reduce training time.
5. Optimize the model's hyperparameters to improve its efficiency.
6. Implement dimensionality reduction techniques like PCA or t-SNE to reduce the number of features.
7. Use more efficient algorithms or models that are specifically designed for faster training and testing.

```
Linear Regression:
Train MAE: 2083.362744368039
Train R2: 0.7622865842301308
Test MAE: 3195.5638434857306
Test R2: 0.8511556012001812
Train Time: 0.007992982864379883 saniye
Test Time: 0.007961511611938477 saniye
Optimistic MAE: 3195.563843486241
Optimistic R2: 0.8511556012001312
```

ORIGINAL DATASET CLASSIFICATION PROJECT

3.5 DATASET AND PREPROCESSING

- **Data source:** Our original dataset from www.twitter.com (from twitter bot we coded before)
- **Data Description:** The dataset columns are as shown below. A dataset with 285 rows and 16 columns. The final state of this data after preprocessing

```
Index(['Id', 'Like Count', 'Comment Count', 'retweet Count', 'View count',
       'Title', 'countOfWords', 'countOfPositive', 'countOfNegative',
       'C_Text__mysteries', 'C_Text__replying', 'T_Text__replying',
       'T_Text__manchester', 'C_Text__manchester', 'T_Text__believe',
       'T_Text__mark'],
      dtype='object')
```

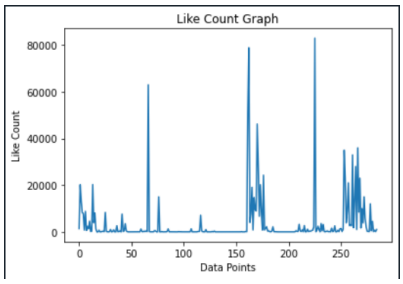
Like Count = Shows how many likes the tweet received.
Comment Count = Shows how many comments the tweet received.
Retweet Count = Shows how many retweet the tweet received.
View Count = Shows the number of times the tweet was viewed .
countOfWords = Shows how many words the tweet consists of.
countOfPositive = Returns the number of positive words in the tweet. This number was determined according to the words in the data set we prepared ourselves.
countOfNegative = Returns the number of negative words in the tweet. This number was determined according to the words in the data set we prepared ourselves.
Title = These column shows what is the tweet about.

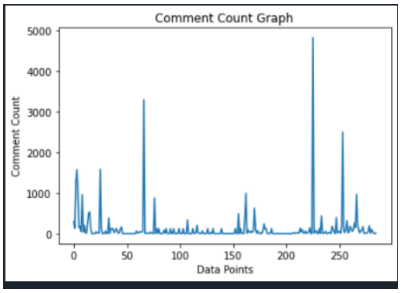
C_Text__mysteries,C_Text__replying,T_Text__replying,T_Text__manchester,C_Text__manchester,T_Text__believe,T_Text__mark = These columns are the columns that make the biggest contribution to the data as a result of the Tf-idf and count vectorization processes of the tweet content

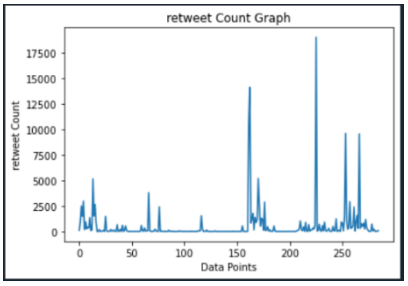
- **ML Problem definition:** Our classification purpose is to find under which "Title" a tweet is shared according to the given characteristics. Our dataset is divided into 5 headings, these are: **Politics, sports, art, science and health.**
- **Data Split:** The division of the dataset into training, validation, and testing sets. 70% of the dataset is reserved for train and 30% for testing.

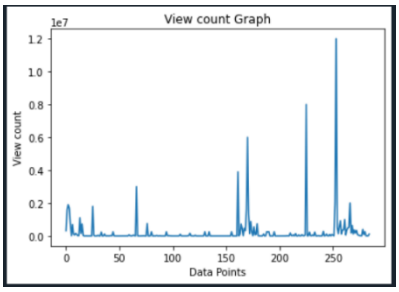
```
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.30, random_state=42)
```

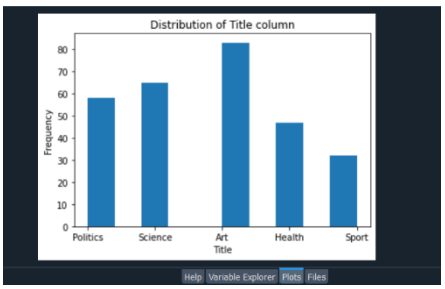
Data Exploration:
Some Columns Hist graphics shown as below, you can see the distribution on datas











- **Data Preprocessing:**

First of all, we read the unprocessed dataset and then applied this steps:

1 -

```
dataset.drop_duplicates(inplace=True)
```

2 - Instead of the K and M values in the comment, retweet and view columns, we convert those columns into numerical data by typing their numerical equivalents.

```
like_column = 'Like Count'
comment_column = 'Comment Count'
retweet_column = 'retweet Count'
view_column = 'View count'

dataset[like_column]

def convert_k_m(column_name):
    for i in range(len(dataset)):
        deger = str(dataset.at[i, column_name])
        if 'K' in deger:
            deger = deger.replace('K', '')
            deger = float(deger) * 1000
            dataset.at[i, column_name] = int(deger)
        elif  'M' in deger:
            deger = deger.replace('M', '')
            deger = float(deger) * 1000000
            dataset.at[i, column_name] = int(deger)

convert_k_m(like_column)
convert_k_m(comment_column)
convert_k_m(retweet_column)
convert_k_m(view_column)
```

3 - Here, we fill in the missing values with the average of the column using the mean() function. That is, the missing values are replaced by the average of the other values in that column.

```
dataset[comment_column].fillna(dataset[comment_column].mean(), inplace=True)
dataset[comment_column] = dataset[comment_column].astype(int)
dataset[like_column].fillna(dataset[like_column].mean(), inplace=True)
dataset[like_column] = dataset[like_column].astype(int)
dataset[view_column].fillna(dataset[view_column].mean(), inplace=True)
dataset[view_column] = dataset[view_column].astype(int)
dataset[retweet_column].fillna(dataset[retweet_column].mean(), inplace=True)
dataset[retweet_column] = dataset[retweet_column].astype(int)
```

4 - We start to infer meaning in the text column. These text-mining steps:

We calculate the number of words in the text content and save it to the countOfWords column and we find the number of positive and negative words in the text content, based on the data set we prepared ourselves last term. And we save them in countOfNegative and countOfPositive columns.

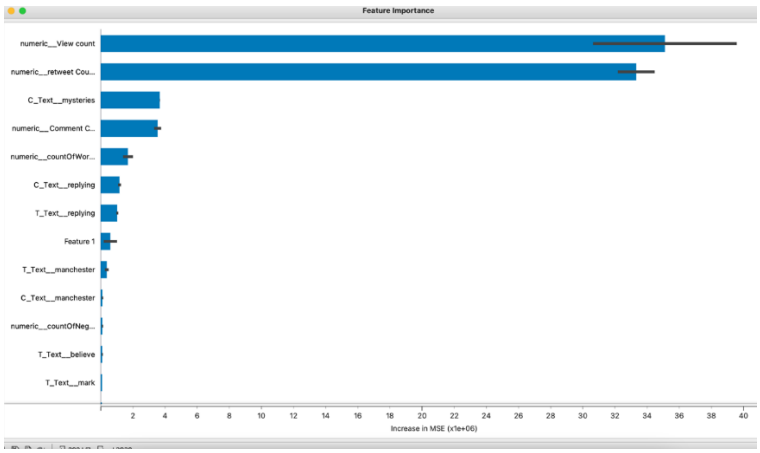
```
text_column = dataset["Text"]
#kelime sayısını ayırma
def countOfWord(text):
    liste = []
    for x in range(0,len(text)):
        liste.append(len(text[x].split()))
    return liste

countOfText = countOfWord(text_column)
dataset["countOfWords"] = countOfText

data_positive = pd.read_csv("PositiveWordsEng.csv")
data_negative = pd.read_csv("NegativeWordsEng.csv")
data_positive = data_positive["PositiveWords"]
data_negative = data_negative["NegativeWords"]

def search_tw(data_words,data_tw):
    liste = []
    for tweet in data_tw:
        count = 0
        tweet = str(tweet).lower().split()
        for word in data_words:
            if word in tweet:
                count+=1
        liste.append(count)
    return liste

countOfPositive = search_tw(data_positive,text_column)
countOfNegative = search_tw(data_negative,text_column)


dataset["countOfPositive"] = countOfPositive

dataset["countOfNegative"] = countOfNegative

dataset.to_csv("dataset_c.csv",index=False)
```

5 - We perform text-mining by applying TfIdf and count vectorixation operations to the text column. Since there are about 4 thousand columns here, we make feature importance on Orange and add the ones that give the best results according to r2 and MSE values to the data set.

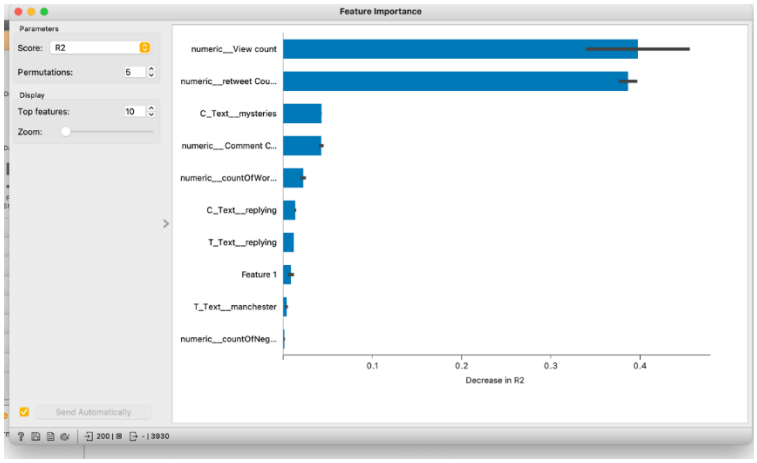
```
# Preprocess text data
Tfidf_transformer = TfidfVectorizer(stop_words=my_stop_words)
count_transformer = CountVectorizer(stop_words=my_stop_words)
scaler = StandardScaler()

# Combine text and numeric features
preprocessor = ColumnTransformer(transformers=[
    ("T_Text", Tfidf_transformer, "Text"),

    ('C_Text', count_transformer, 'Text'),
    ('numeric', 'passthrough', coulmn_names)
])
# Fit and transform the preprocessor to the data
X_preprocessed = preprocessor.fit_transform(X)
```

The values in Orange;

MSE:



R2 score:

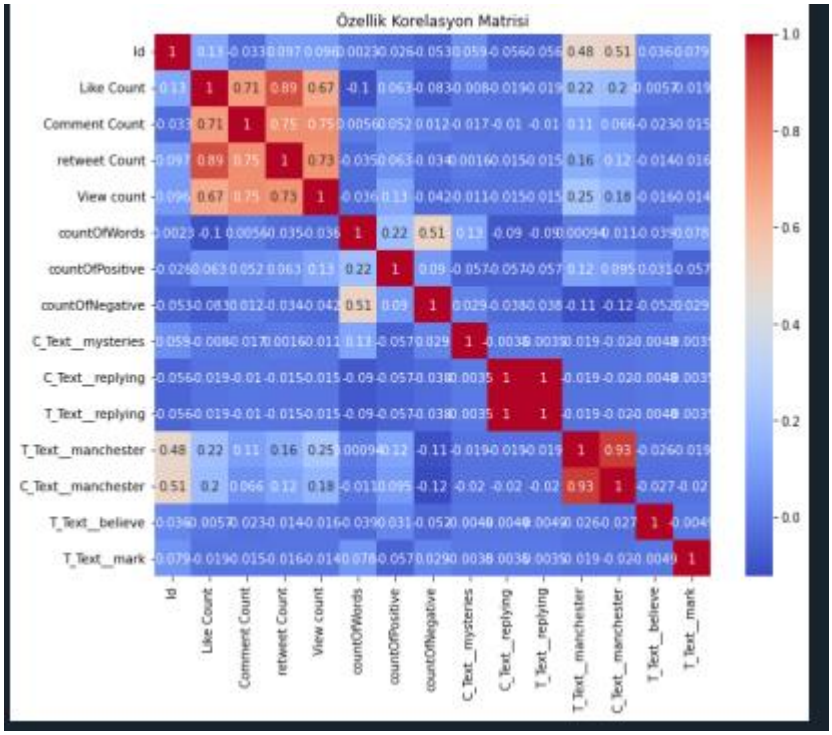


6 - We are deleting the user_name and date of tweet columns that do not contribute to the data set. Since we have completed our operations in the Text column, we delete it as well. Finally, if there are duplicate values, we delete them and remove the null values from the data set.

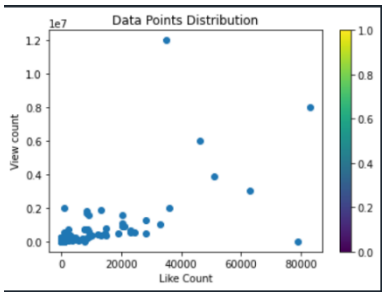
```
dataset = pd.read_csv("dataset_c.csv")
dataset = dataset.drop("Text",axis=1)
dataset = dataset.drop("User Name",axis=1)
dataset = dataset.drop("Date of Tweet",axis=1)

dataset.to_csv("dataset_c_noText.csv",index=False)
dataset_noText = dataset = pd.read_csv("dataset_c_noText.csv")


merged_df = pd.merge(dataset_noText, text_df, on='Id')
merged_df.to_csv("cl_dataset.csv",index=False)

data_new = pd.read_csv("cl_dataset.csv")
```

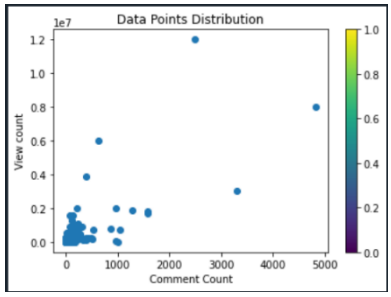
- **Data Usage:** Machine learning algorithms such as AdaBoost, Gradient Boosting, kNN, Random Forest will be used. According to these algorithms, Tweet content will be predicted.

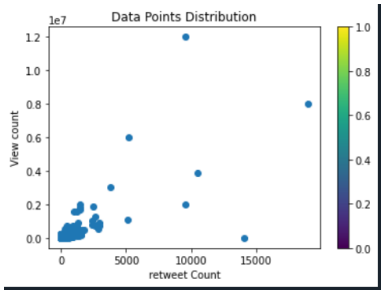
- **Feature Engineering:** Since we use our own dataset, the correlation value of most of our features is close to 0, but the correlation value between tweet likes, comments, retweets, and views is close to 1.

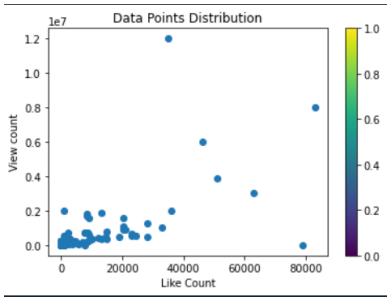


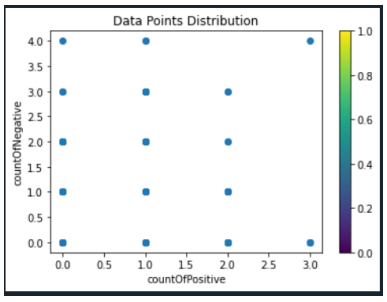
- **Dataset visualization:** We did the data distribution analysis for some features.











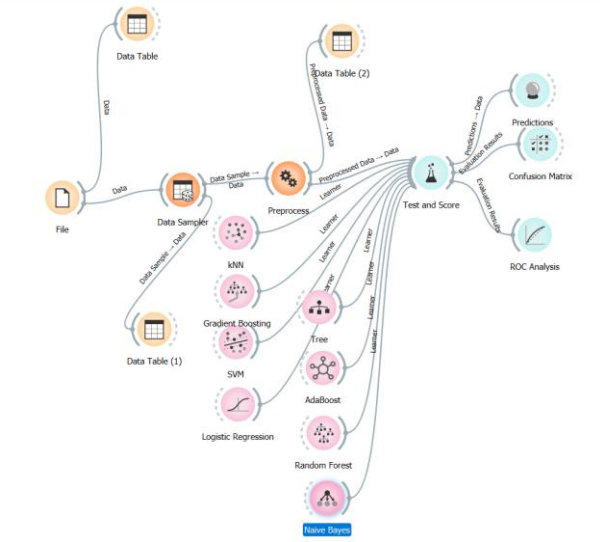
3.6 CLASSIFICATION STEPS

- **Data Collection:** We collected our data from Twitter.
- **Data Cleaning:** We removed duplicate data, nulls, outliers that corrupted the dataset.
- **Feature selection:** We determined the features in the data set, we added some features such as the total number of words, the number of positive words, the number of negative words according to the text content, and some features by doing extra text mining.
- **Data Splitting:** We split the dataset into two for training and testing.
- **Model Selection and Training:** At the beginning we tried to find the best model using the Orange application, then we coded it.
- **Model development:** We adjusted the hyperparameters or algorithm configurations to improve the performance of the model. At this stage, we identified issues such as overfitting or underfitting.

3.7 CLASSIFICATION RESULTS AND MODEL SELECTION

First of all, we tried to find the best model in Orange. According to the program results, the best model was **AdaBoost**.

Other model's results given below:

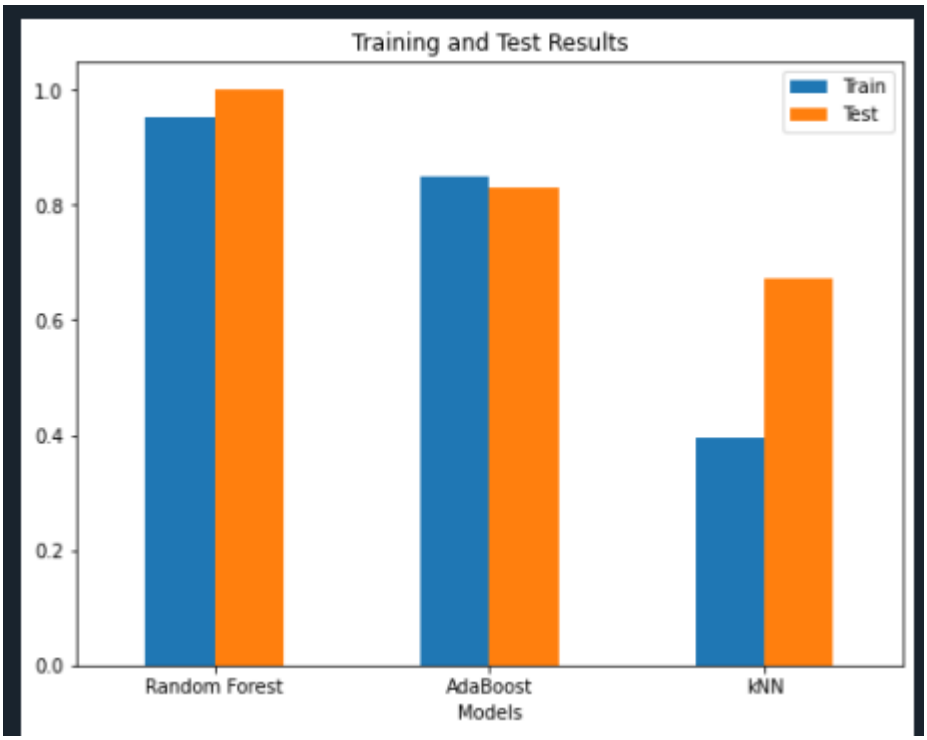


Model	AUC	CA	F1	Precision	Recall
AdaBoost	0.997	0.996	0.996	0.996	0.996

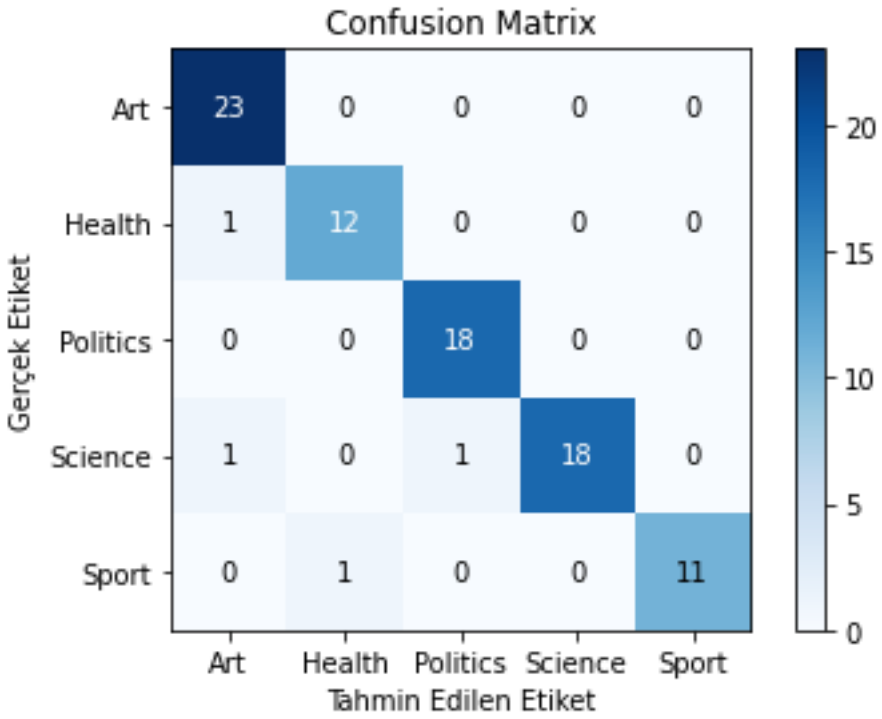
Model	AUC	CA	F1	Precision	Recall
Random Forest	0.993	0.912	0.912	0.915	0.912
Logistic Regression	0.974	0.860	0.860	0.863	0.860

Model	AUC	CA	F1	Precision	Recall
kNN	0.943	0.811	0.811	0.815	0.811
Naive Bayes	0.952	0.789	0.787	0.788	0.789

- Comparing training and test results: When we coded the models which are Random Forest, Adaboost,and the best result was **Random Forest.**



Confusion matrix: for model **Random Forest:**



3.8 CLASSIFICATION HYPERPARAMTER OPTIMIZATION RESULTS

- Use best training algorithm and try to tune hyperparameters to get better performance metrics.
- Try to reduce test and training time.
- Try to improve performance metric.

Our best training result was 1, so we didn't try to better train the best model. Instead, we tried to make another model better which is Adaboost.

We changed the hyper parameters of the model and made the performance of the model better. You can see the best hyper parameter values below. Adaboost training accuracy value increased from **0.8291457286432161** to **0.9748717948717948**

```
kNN Training Accuracy: 0.6733668341708543
kNN Test Accuracy: 0.3953488372093023
AdaBoost Training Accuracy: 0.8291457286432161
AdaBoost Test Accuracy: 0.8488372093023255
Random Forest Training Accuracy: 1.0
Random Forest Test Accuracy: 0.9534883720930233
Adaboost Best Parameters: {'base_estimator__max_depth': 3, 'learning_rate': 0.1, 'n_estimators': 50}
Adaboost Best Training Accuracy: 0.9748717948717948
```

Train and Test Time: We have not a big dataset so the time is good. But for huge dataset, we can do for reducing the time these:

8. Reduce the size of the dataset by selecting a smaller subset of data.
9. Perform feature selection or feature extraction to eliminate irrelevant or redundant features.
10. Utilize parallel processing techniques such as multi-threading or multiprocessing to train and test the model simultaneously.
11. Apply techniques like random under-sampling or random over-sampling to balance the dataset and reduce training time.
12. Optimize the model's hyperparameters to improve its efficiency.
13. Implement dimensionality reduction techniques like PCA or t-SNE to reduce the number of features.
14. Use more efficient algorithms or models that are specifically designed for faster training and testing.

```
Train Time: 0.10000038146972656 second
Test Time: 0.020999908447265625 second
```

ORIGINAL DATASET CLUSTERING PROJECT

3.9 DATASET AND PREPROCESSING

- **Data Source:** www.twitter.com
- **Data Description:**

```
RangeIndex: 284 entries, 0 to 283
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Id                  284 non-null    int64
 1   Like Count          284 non-null    int64
 2   Comment Count       284 non-null    int64
 3   retweet Count       284 non-null    int64
 4   View count          284 non-null    int64
 5   countOfWords        284 non-null    int64
 6   countOfPositive     284 non-null    int64
 7   countOfNegative     284 non-null    int64
 8   Art                 284 non-null    int64
 9   Health              284 non-null    int64
 10  Politics            284 non-null    int64
 11  Science             284 non-null    int64
 12  Sport               284 non-null    int64
 13  C_Text__mysteries   284 non-null    float64
 14  C_Text__replying    284 non-null    float64
 15  T_Text__replying    284 non-null    float64
 16  T_Text__manchester  284 non-null    float64
 17  C_Text__manchester  284 non-null    float64
 18  T_Text__believe     284 non-null    float64
 19  T_Text__mark        284 non-null    float64
dtypes: float64(7), int64(13)
```

A dataset with 284 rows and 19 columns. The final state of this data after preprocessing
Like Count = Shows how many likes the tweet received
Comment Count = Shows how many comments the tweet received
Retweet Count = Shows how many retweet the tweet received
View Count = Shows the number of times the tweet was viewed
countOfWords = Shows how many words the tweet consists of
countOfPositive = Returns the number of positive words in the tweet. This number was determined according to the words in the data set we prepared ourselves.
countOfNegative = Returns the number of negative words in the tweet. This number was determined according to the words in the data set we prepared ourselves.
Art,Health,Politics,Science,Sport = These columns show which sector the tweet is in, for example, if the tweet is in the Arts sector, it would say 1 if not 0.
C_Text__mysteries,C_Text__replying,T_Text__replying,T_Text__manchester,C_Text__manchester,T_Text__believe,T_Text__mark = These columns are the columns that make the biggest contribution to the data as a result of the Tf-idf and count vectorization processes of the tweet content.

- **ML Problem definition:** The purpose of this cluster method is to find outlier values in the data set and to determine how much these values increase or decrease the accuracy of the data set.

- **Data Split:** The division of the dataset into training, validation, and testing sets.

```
#########################################################
#train test split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df_scaled,y, test_size=0.30, random_state=42)
```

70% of the dataset is reserved for train and 30% for testing.

- **Data Exploration:**
The raw states of the Data Set after we took it ourselves.
First 2 rows of dataset

Id	Text	User Name	Date of Tweet	Like Count	Comment Coun	retweet Count	View count	Title
0	Donald Trump made this disturbing post on his social media app Truth Social: "WORLD WAR III"	DailyCount	4h	1418	292	116	317K	Politics
1	No way jidion met Donald Trump at ufc 287	shannonsharpeee	18h	20.2K	134	1014	1.6M	Politics

Data size is 286 rows and 9 columns.

```
(286, 9)
```

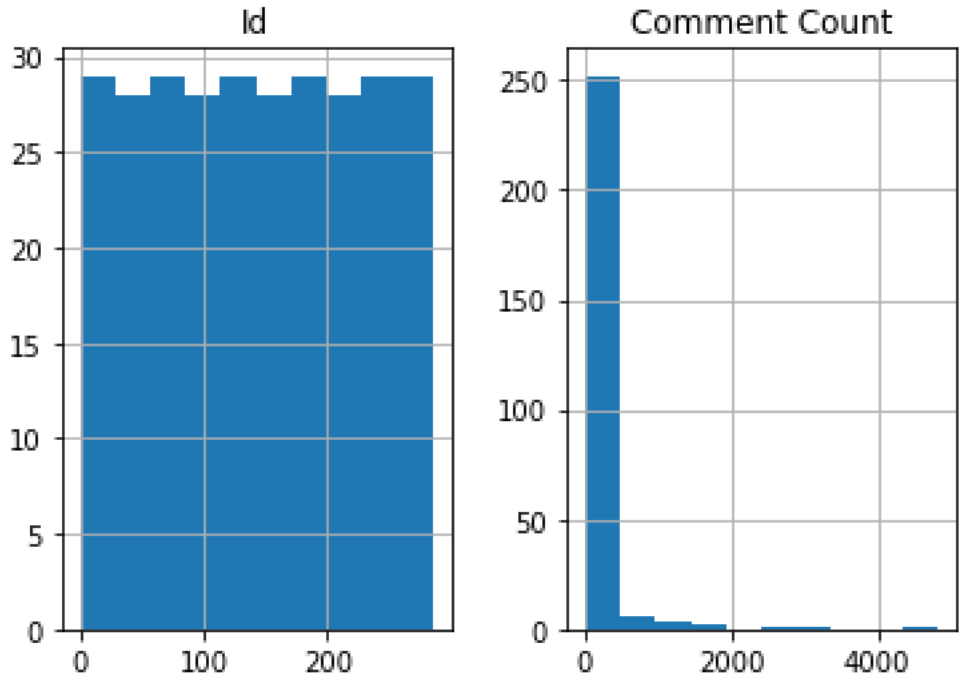
Data types of columns;

```
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             286 non-null    int64
 1   Text           286 non-null    object
 2   User Name      285 non-null    object
 3   Date of Tweet  284 non-null    object
 4   Like Count     285 non-null    object
 5   Comment Count  267 non-null    float64
 6   retweet Count  284 non-null    object
 7   View count     276 non-null    object
 8   Title          286 non-null    object
dtypes: float64(1), int64(1), object(7)
```

There are 2 numeric values in the columns. The “m”, “k” texts in the like, retweet, view columns should be removed and made numerical in them.

Hist values;

Since there are only 2 numeric values, two figures appear. Here, we can reach the information that a maximum of 0-2000 comments were made.



When we look at the unique values of the title column, we see 5 different title values.

```
['Politics' 'Science' 'Art' 'Health' 'Sport']
```

- **Data Preprocessing:**

Steps:

1) Instead of the K and M values in the comment, retweet and view columns, we convert those columns into numerical data by typing their numerical equivalents.

```
like_column = 'Like Count'
comment_column = 'Comment Count'
retweet_column = 'retweet Count'
view_column = 'View count'

dataset[like_column]

def convert_k_m(column_name):
    for i in range(len(dataset)):
        deger = str(dataset.at[i, column_name])
        if 'K' in deger:
            deger = deger.replace('K', '')
            deger = float(deger) * 1000
            dataset.at[i, column_name] = int(deger)
        elif  'M' in deger:
            deger = deger.replace('M', '')
            deger = float(deger) * 1000000
            dataset.at[i, column_name] = int(deger)

convert_k_m(like_column)
convert_k_m(comment_column)
convert_k_m(retweet_column)
convert_k_m(view_column)
```

2) We start to infer meaning in the text column. These text-mining steps

(1) We calculate the number of words in the text content and save it to the countOfWords column

```
def countOfWord(text):
    liste = []
    for x in range(0,len(text)):
        liste.append(len(text[x].split()))
    return liste

countOfText = countOfWord(text_column)
```

(2) We find the number of positive and negative words in the text content, based on the data set we prepared ourselves last term. And we save them in countOfNegative and countOfPositive columns.

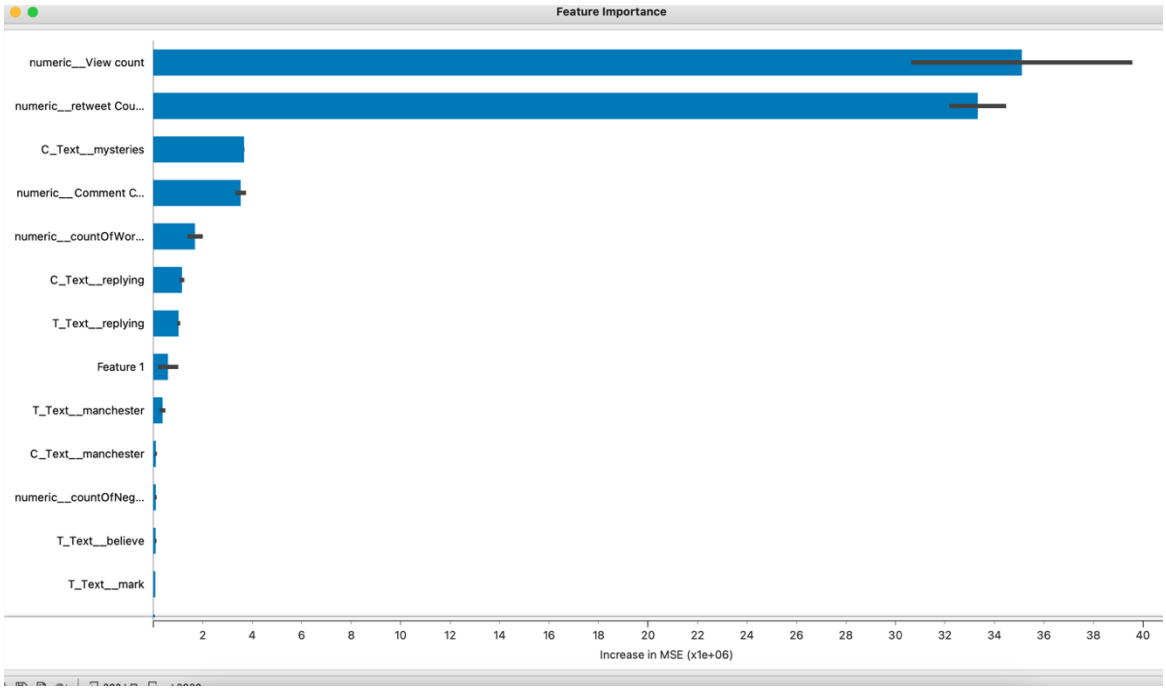
```
data_positive = pd.read_csv("PositiveWordsEng.csv")
data_negative = pd.read_csv("NegativeWordsEng.csv")
data_positive = data_positive["PositiveWords"]
data_negative = data_negative["NegativeWords"]

def search_tw(data_words,data_tw):
    liste = []
    for tweet in data_tw:
        count = 0
        tweet = str(tweet).lower().split()
        for word in data_words:
            if word in tweet:
                count+=1
        liste.append(count)
    return liste

countOfPositive = search_tw(data_positive,text_column)
countOfNegative = search_tw(data_negative,text_column)
```

(3) We perform text-mining by applying TfIdf and count vectorixation operations to the text column. Since there are about 4 thousand columns here, we make feature importance on Orange and add the ones that give the best results according to r2 and MSE values to the data set.

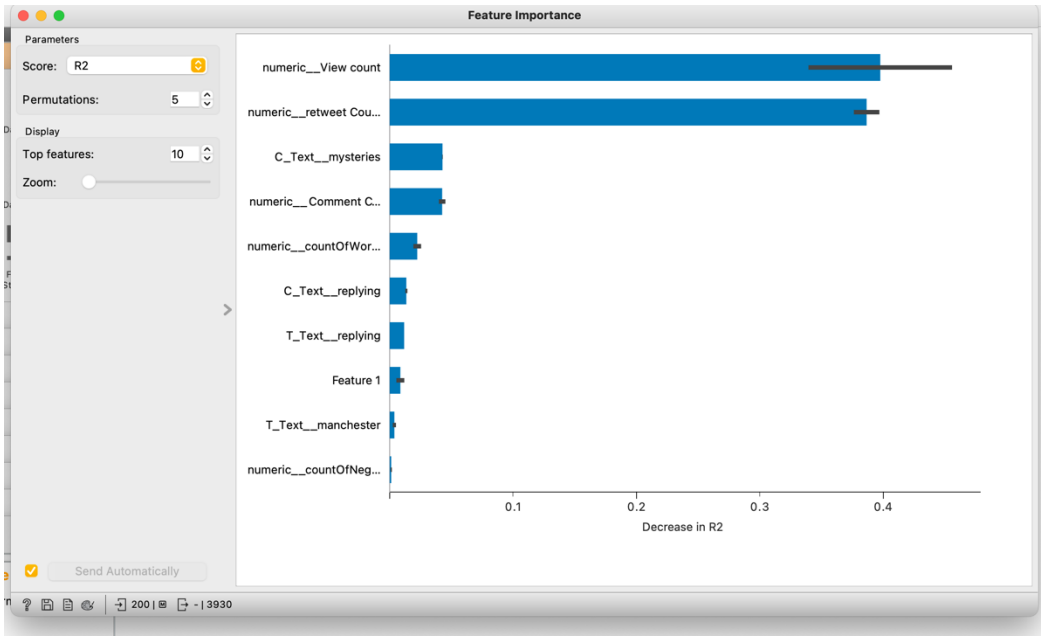
```
# Preprocess text data
Tfidf_transformer = TfidfVectorizer(stop_words=my_stop_words)
count_transformer = CountVectorizer(stop_words=my_stop_words)
scaler = StandardScaler()

# Combine text and numeric features
preprocessor = ColumnTransformer(transformers=[
    ("T_Text", Tfidf_transformer, "Text"),

    ('C_Text', count_transformer, 'Text'),
    ('numeric', 'passthrough', coulmn_names)
])
# Fit and transform the preprocessor to the data
X_preprocessed = preprocessor.fit_transform(X)
```

(4)

The values in Orange;
MSE



R2 score:



3) We are deleting the user_name and date of tweet columns that do not contribute to the data set. Since we have completed our operations in the Text column, we delete it as well.

```
dataset = dataset.drop("Text",axis=1)
dataset = dataset.drop("User Name",axis=1)
dataset = dataset.drop("Date of Tweet",axis=1)
```

4) We apply one-hot-encoding operations to the title column, because our goal is to find outlier values, so object values will not work for us.

```
one_hot_df = pd.get_dummies(dataset['Title'])
dataset['ID'] = range(1, len(dataset) + 1)
one_hot_df['ID'] = range(1, len(one_hot_df) + 1)


one_hot_df[one_hot_df == True] = "1"
one_hot_df[one_hot_df == False] = "0"

merged_df = pd.merge(dataset, one_hot_df, on='ID')
```

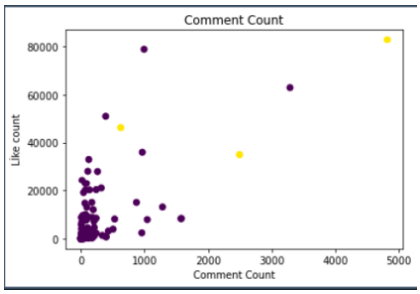
5) Finally, if there are duplicate values, we delete them and remove the null values from the data set.

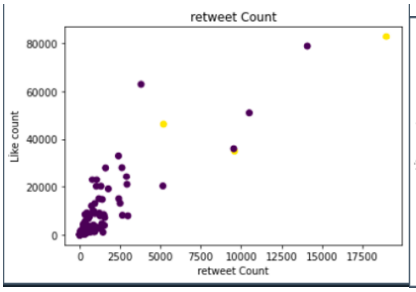
```
dataset.dropna(inplace=True)
```

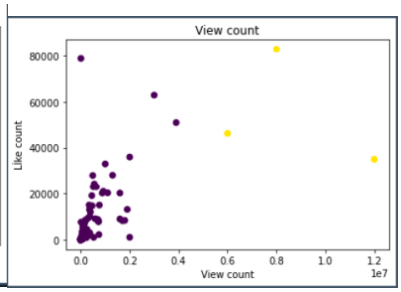
```
dataset.drop_duplicates(inplace=True)
#pd.read_excel("all_tweets.xlsx")
```

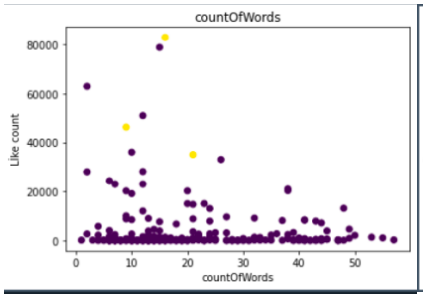
- **Data Usage:**
  DBSCN, K-MEAN and GaussianMixture methods will be used and after outlier extraction with them, the change of outlier extracted data set and outlier unallocated data set will be observed. Evaluation scores will be silhoutte and Davies scores.

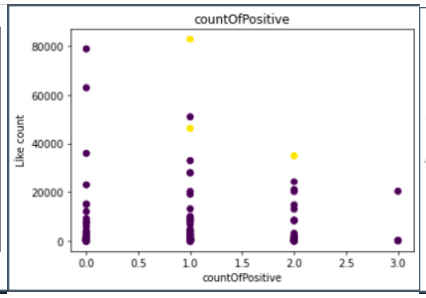
- **Feature engineering:**

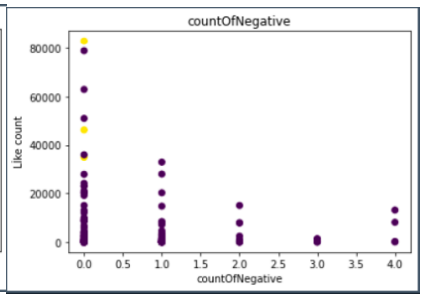


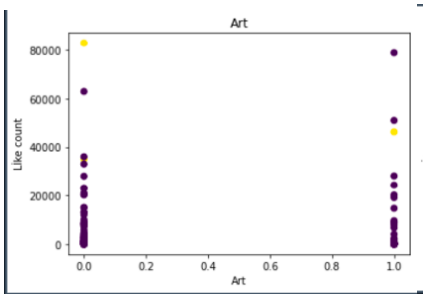


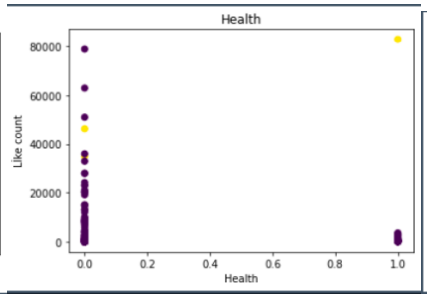


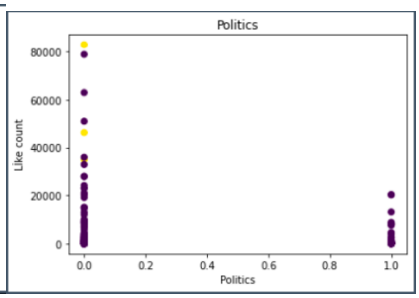


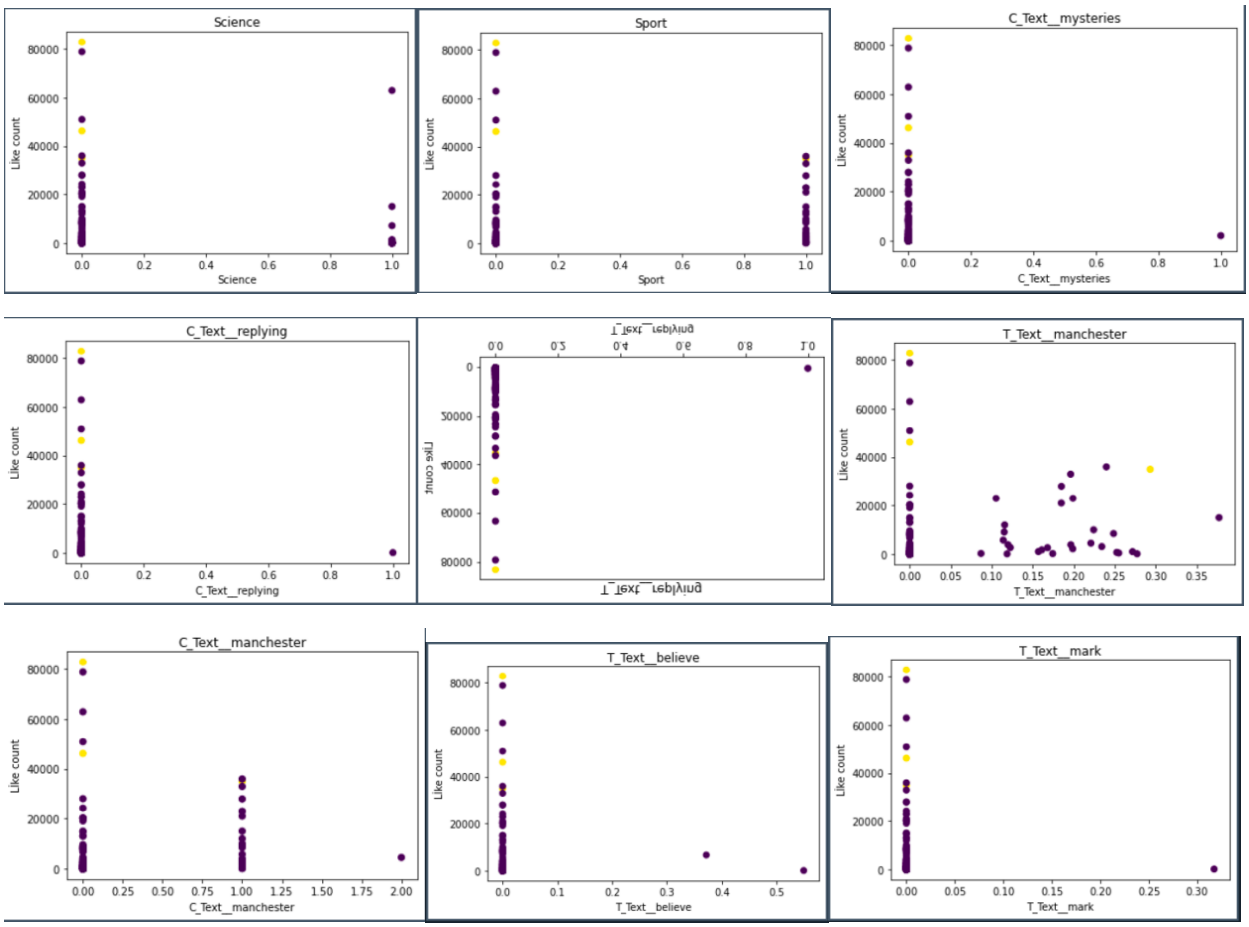










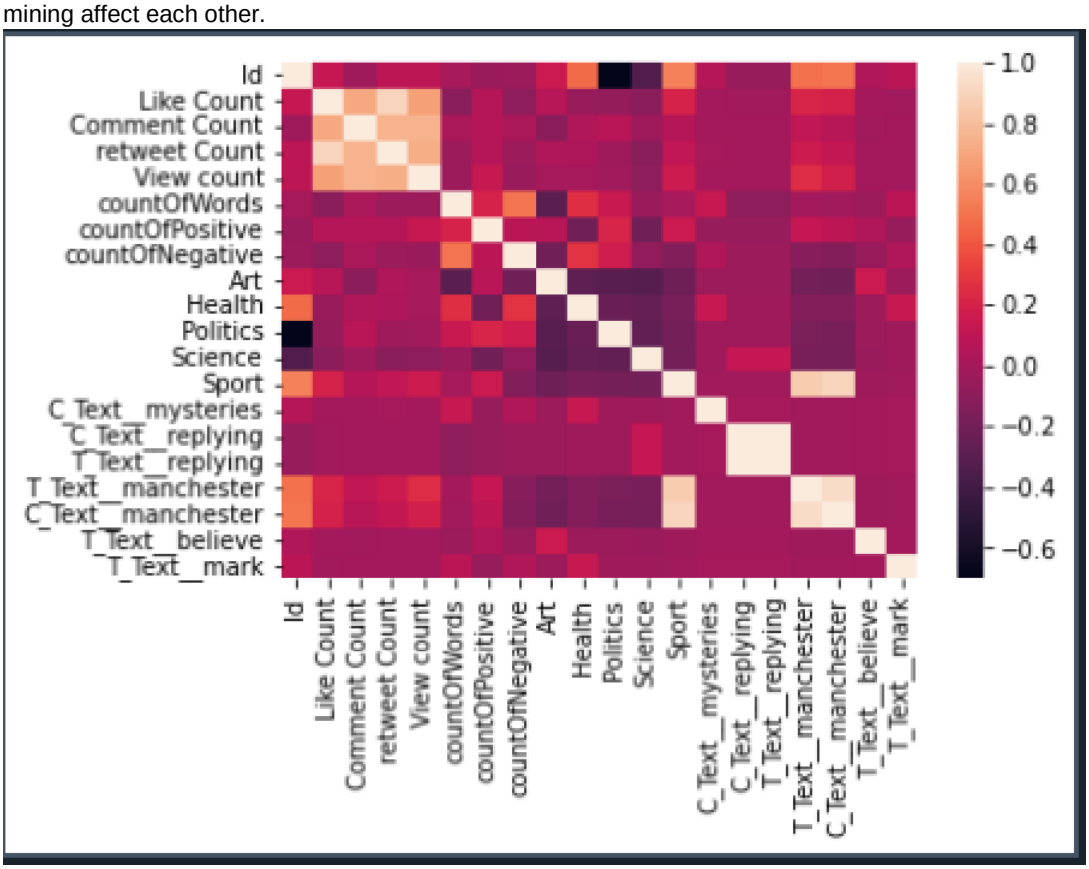


In these graphs, we see the effect of the features on outlier extraction with the K-Means algorithm. For example, as the value in the View Count column increases, the outlier values increase. Or, when the number of words in the countOfWords column is between 10 and 30, the outlier values show redundancy here. If there are no negative words in the CountOfNegative column, we can make inferences that outlier values are found here.

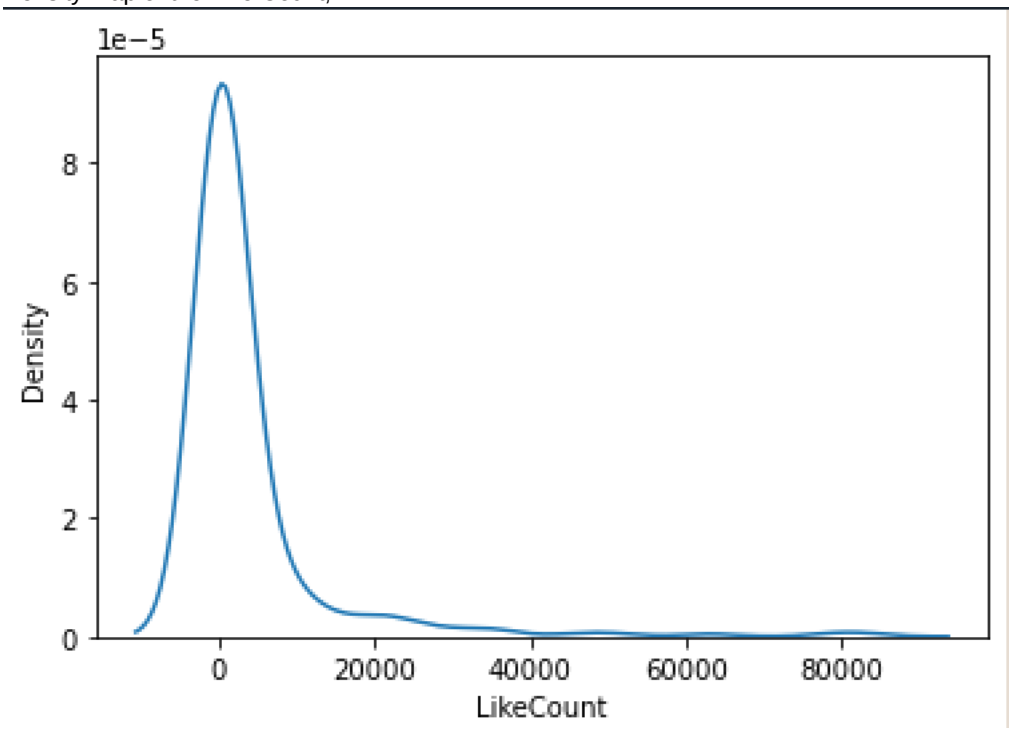
- **Dataset visualization:**

  Correlation graph, with this graph, it is seen that the columns that affect each other the most are like count, comment column, retweet column and view column. At the same time, the columns obtained as a result of text-

mining affect each other.

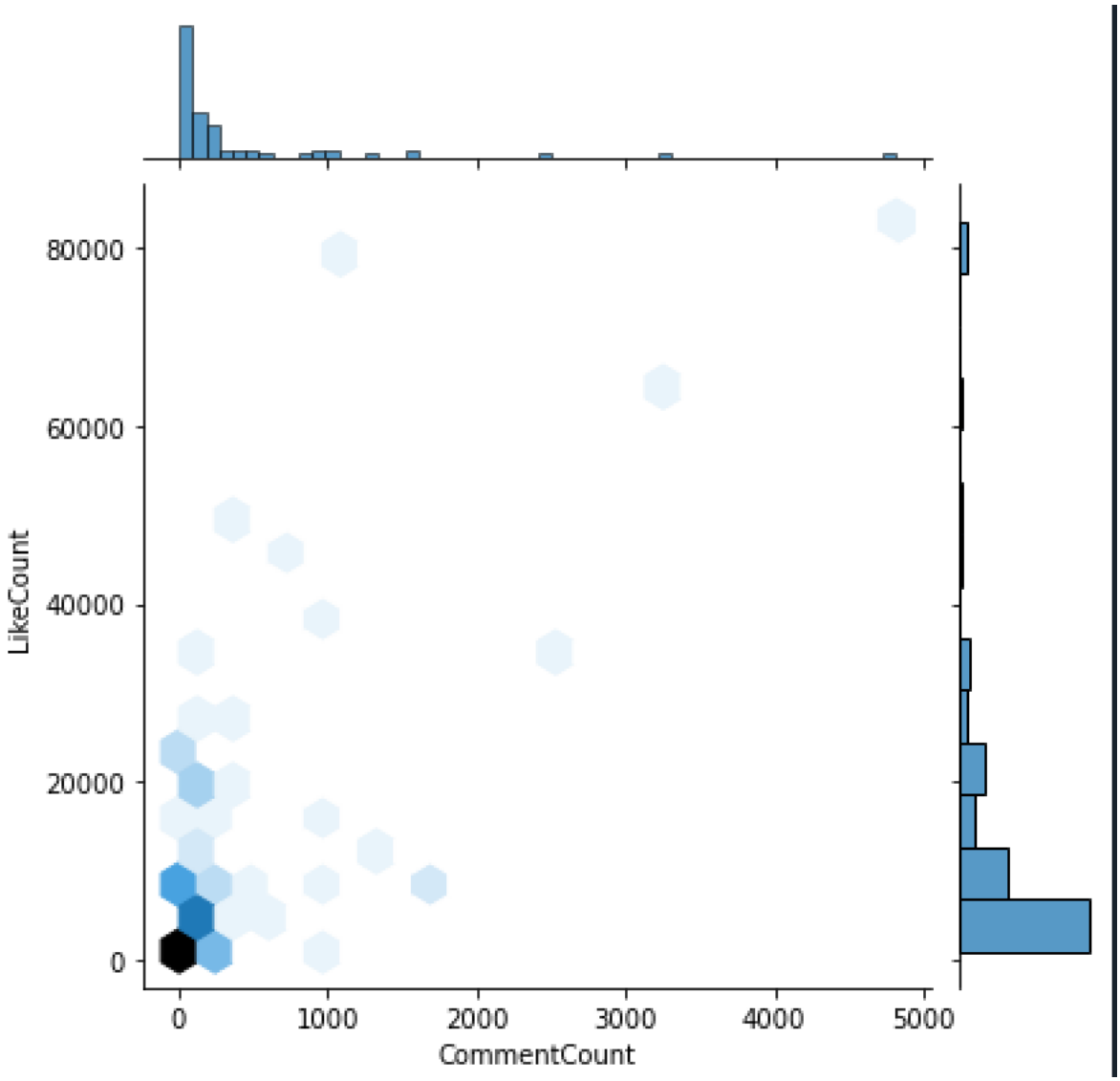


Density map of the Like Count;



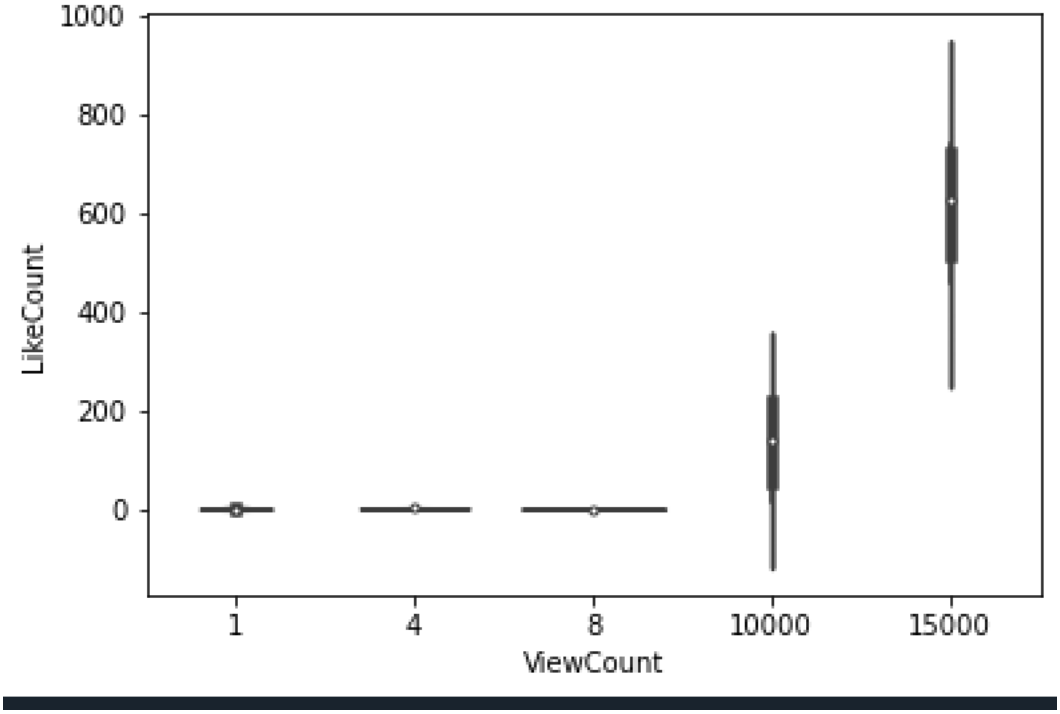
Here, it can be seen that the density of Likes is between 0 and 20000.

Heat map and plot display between Like count and Commet count, where the correlation between them is high;



Here we can see that low number of likes get more comment, this like count is valid for 1000 and below.

The violinplot representation between the number of Likes and the number of views, where the correlation between them is high;



This chart visualizes the distribution of the "CommentCount" and "LikeCount" properties over samples containing the top 5 most frequent values of the "CommentCount" property..

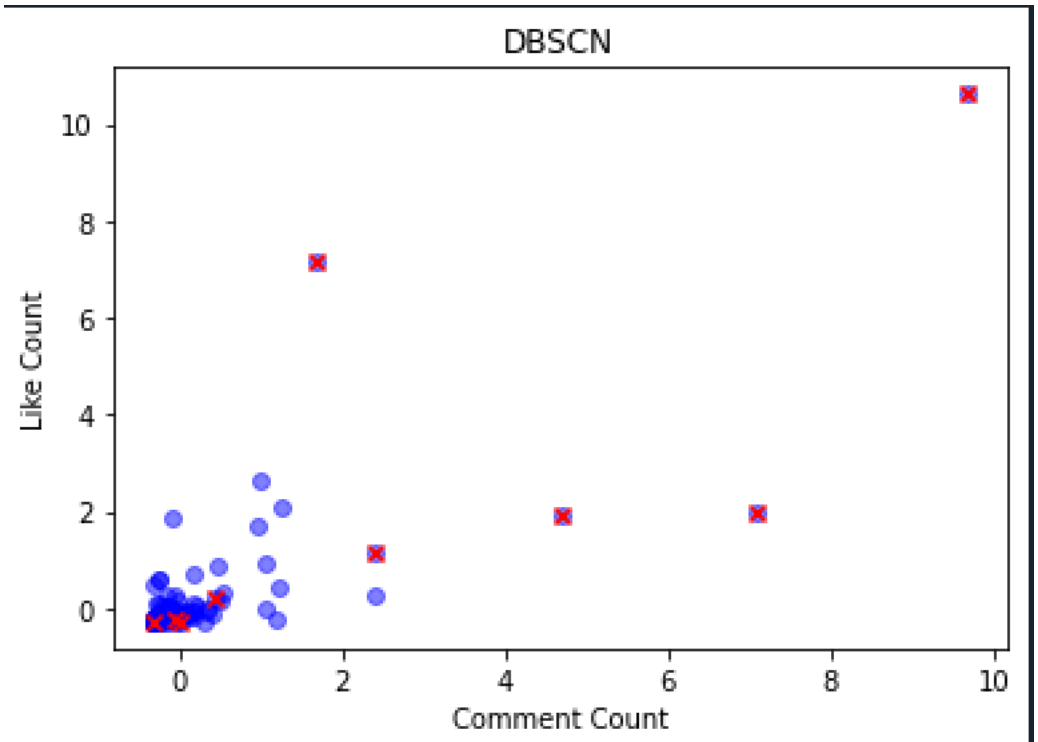
3.10 CLUSTERING STEPS

1. First of all, a standard scaler operation was applied to the dataset because the data distribution was very common.
2. The dataset is split into train test (70-30). Here, the value of y is the number of Likes, the x dataset contains other features apart from the number of likes.
3. DBSCN, K-Means and GMM algorithms were set up to try different algorithms.
4. Best parameters for DBSCN found via Orange eps=2.246 min_samples=2 (2 selected to find outlier)
5. Since those that do not belong to any cluster for DBSCN are shown with -1, the rows with the value of -1 are removed from the dataset and a data set with outlier values is obtained for dbsnc.
6. n_cluster=2 was selected for the K-Means algorithm (to find outlier values)
7. The normal distribution threshold was performed to find outliers for K-Means. If the distance value for each value was greater than the threshold value, the outlier value was accepted. By subtracting these values from the data set, a data set with outlier values for k-Mean was obtained.
8. Silhouette scores of different parameters for GMM were calculated and the parameters that gave the best results were selected.
9. Selected n_components=2 covarience_type=" spherical" for GMM.
10. Normal distiribution was applied to find outlier values in GMM.
11. Outliers above threshold 3 are accepted for GMM. (Z-Score). By subtracting these values, a data set without outliers was obtained for GMM.
12. Outliers extracted data sets were fitted and silhouette and Davies scores were calculated.
13. In the data set (X_test) without outlier values, fit and silhouette and Davies scores were calculated.
14. Evaluations were made in the clusters prepared with 2 different methods of these 3 different cluster methods.

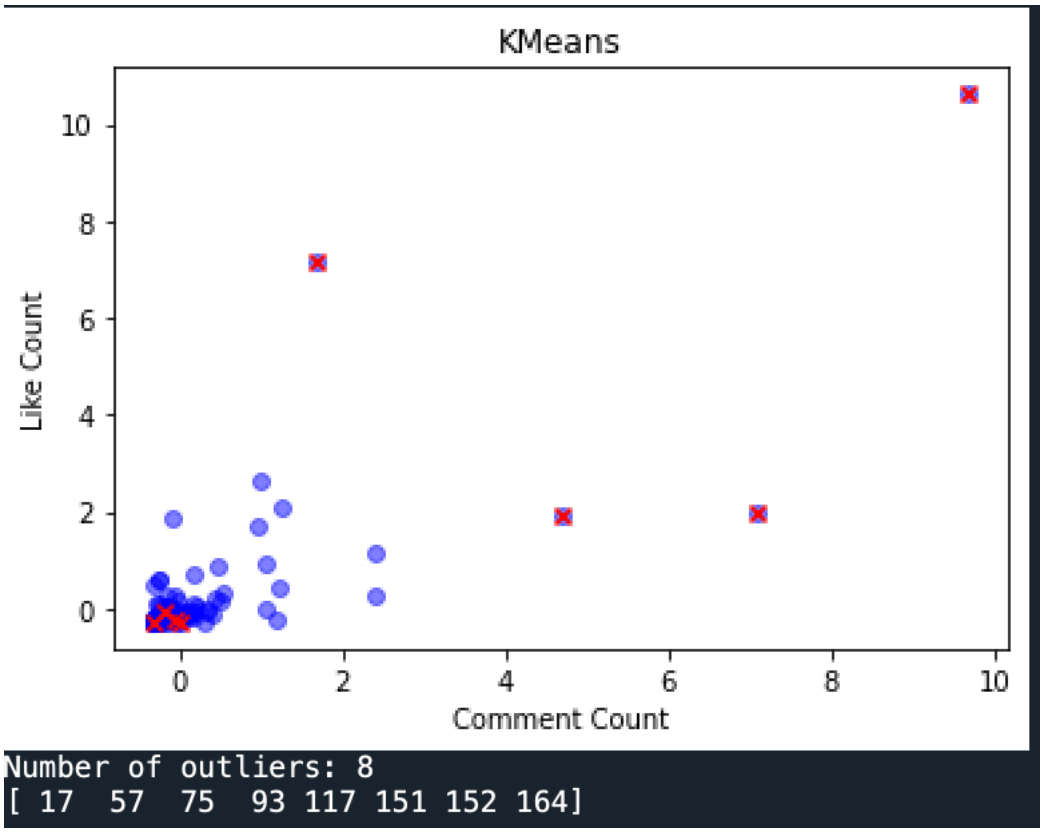
3.11 CLUSTERING RESULTS AND MODEL SELECTION

Three different algorithms were used for the cluster: DBSCN, K-Means and GaussianMixture algorithms. The location and visualization of the outlier values found by these algorithms are given below.

DBSCN;

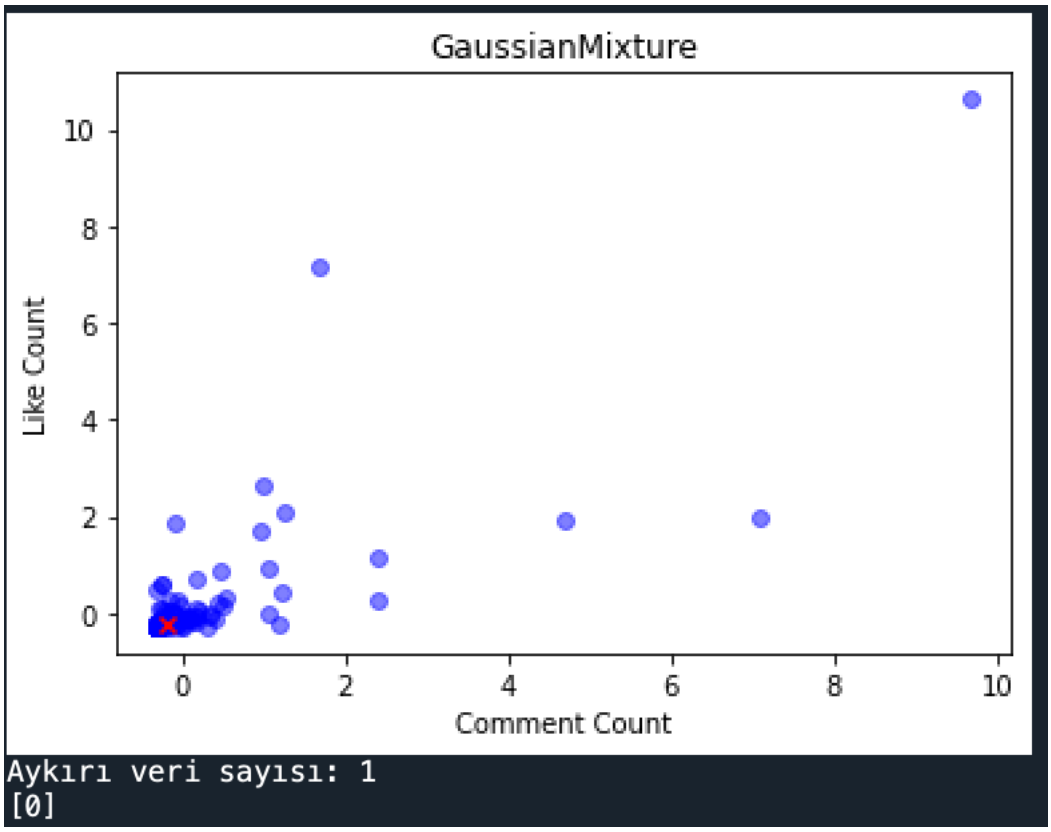


K-Means;




```
Number of outliers: 8
[ 17  57  75  93 117 151 152 164]
```

GMM;

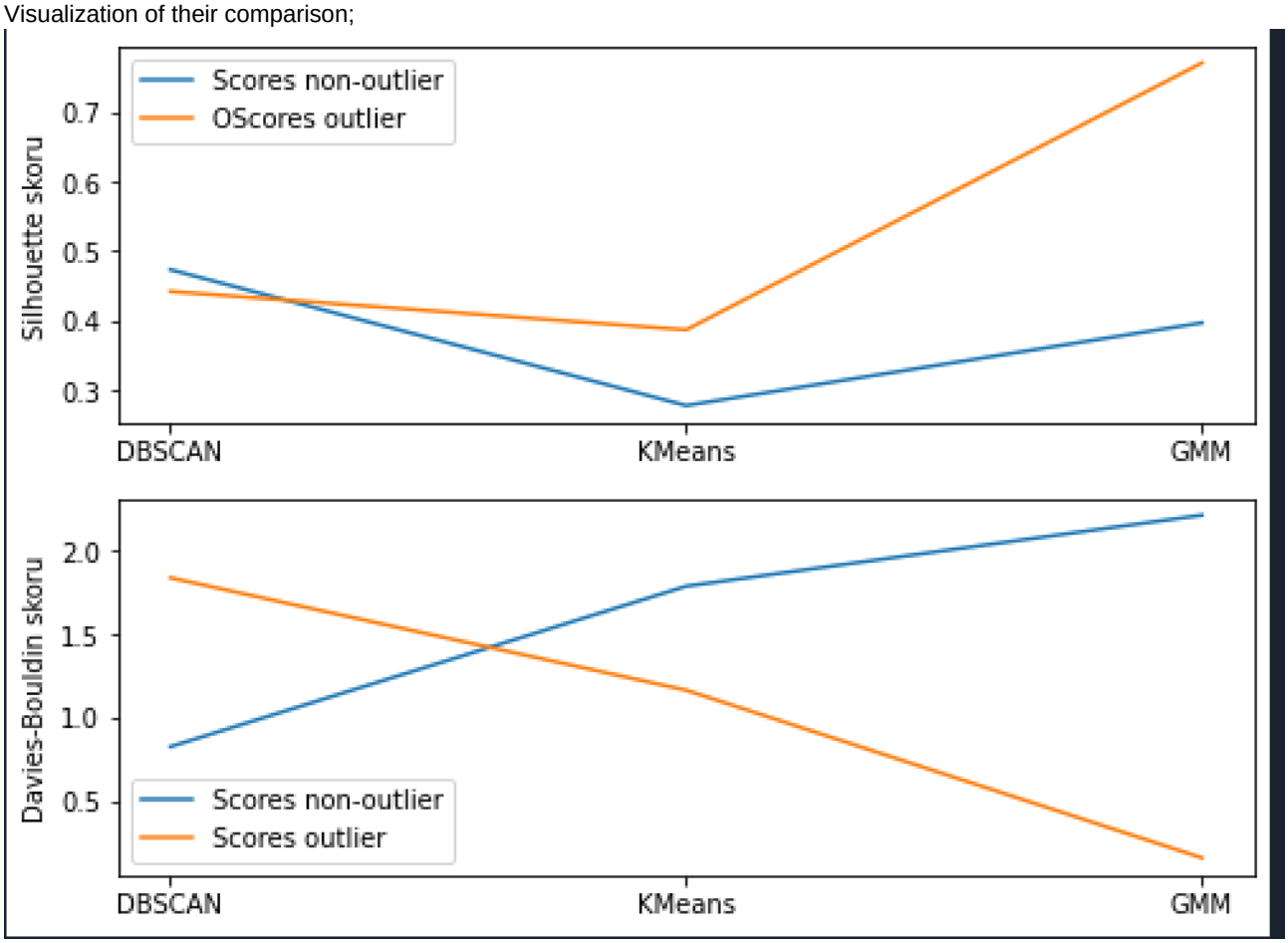


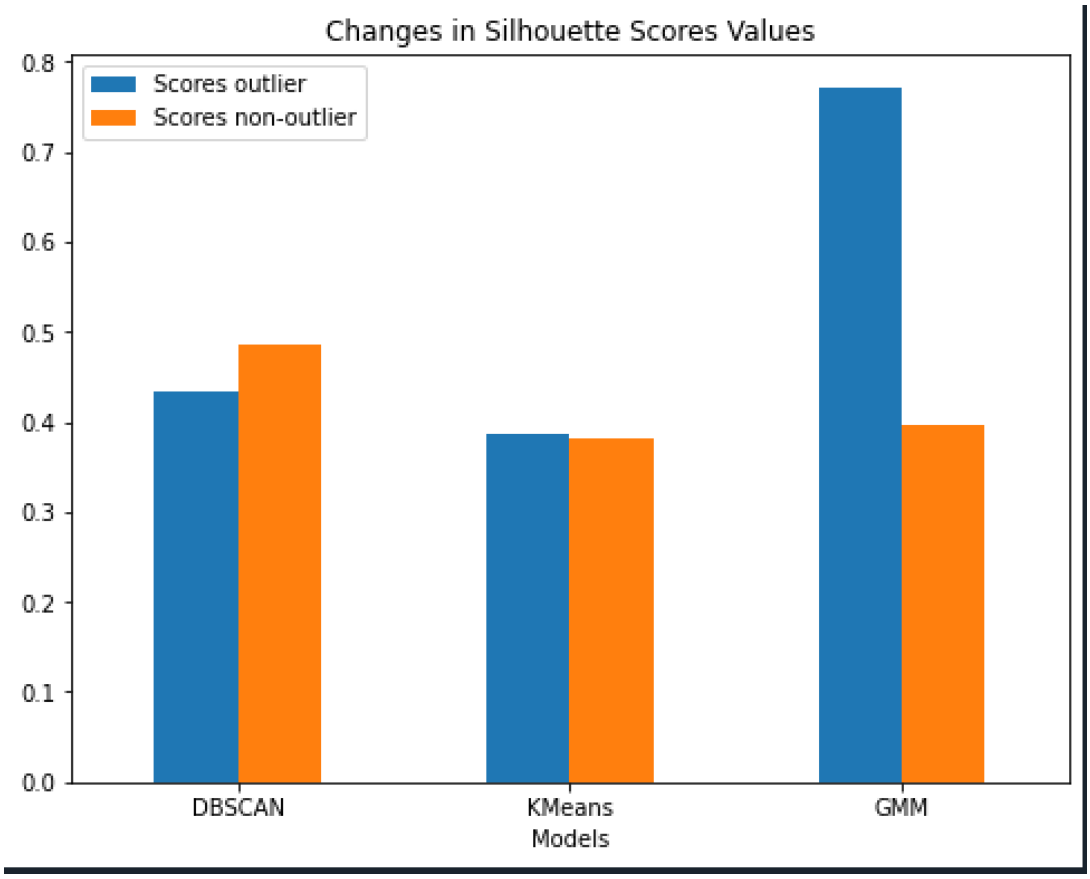

```
Aykırı veri sayısı: 1
[0]
```

Their performance difference between outlier and non-outlier datasets; Here train dataset is not outlier, test dataset is outlier dataset.

```
Scores non-outlier

Silhouette score dbscn nonoutlier:  0.48572030811677946
Silhouette score kmean nonoutlier:  0.38050738642379256
Silhouette score gmm nonoutlier: 0.39658785730975155
Davies score k mean nonoutlier:  1.0245922517438983
Davies score dbscn nonoutlier:  0.7613029162328417
Davies score gmm nonoutlier :  2.2113903757445033



Scores outlier

Silhouette score dbscn outlier 0.4323568378275871
Silhouette score kmean outlier:  0.3865528998888909
Silhouette score gmm outlier: 0.7704501116325587
Davies score k mean outlier:  1.1619401844581363
Davies score dbscn outlier:  1.9398553057536008
Davies score gmm outlier:  0.15669629471653915
```

Visualization of their comparison;





As we can see from these graphs, when we subtract outlier values in the DBSCN algorithm, it gives better results.

But the GMM algorithm gives the best result with 0.77 without removing the outlier values.

3.12 CLUSTERING HYPERPARAMTER OPTIMIZATION RESULTS

The best working algorithm is the GMM algorithm. The silhouette scores that change when we change the parameters of this algorithm sequentially are given below. Accordingly, since we are scanning outliers, the best covariance_type is "spherical" with n_components=2.

Covariance Type	o	Silhouette Score	Davies Bouldin Score	Number ofCluster
full	…	0.423218	0.636979	9
tied	…	0.423218	0.636979	9
diag	…	0.415461	0.870392	7
diag	…	0.414621	0.685985	8
full	…	0.413177	0.841948	7
tied	…	0.413177	0.841948	7
spherical	…	0.412152	2.24083	2
full	…	0.410958	0.698729	8
tied	…	0.410958	0.698729	8
full	…	0.408992	0.89099	14
full	…	0.406383	0.758471	11
tied	…	0.406098	0.803009	12