Project Name:- Malignant-Comments-Classifier.

Importing Libraries

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
```

Loading Dataset

In [6]:

```
import pandas as pd
df=pd.read_csv('train.csv')
df.head()
```

Out[6]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

In [82]:

```
df
```

Out[82]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

In [88]:

```
df.shape
```

Out[88]: (159571, 8)

In [7]:

```
df["malignant"].unique()
```

Out[7]: array([0, 1], dtype=int64)

In [9]:

```
df["highly_malignant"].shape
```

Out[9]: (159571,)

Checking Null Values

In [10]:

```
df.isnull()
```

Out[10]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
159566	False	False	False	False	False	False	False	False
159567	False	False	False	False	False	False	False	False
159568	False	False	False	False	False	False	False	False
159569	False	False	False	False	False	False	False	False
159570	False	False	False	False	False	False	False	False

159571 rows × 8 columns

In [11]:

```
df.isnull().sum()
```

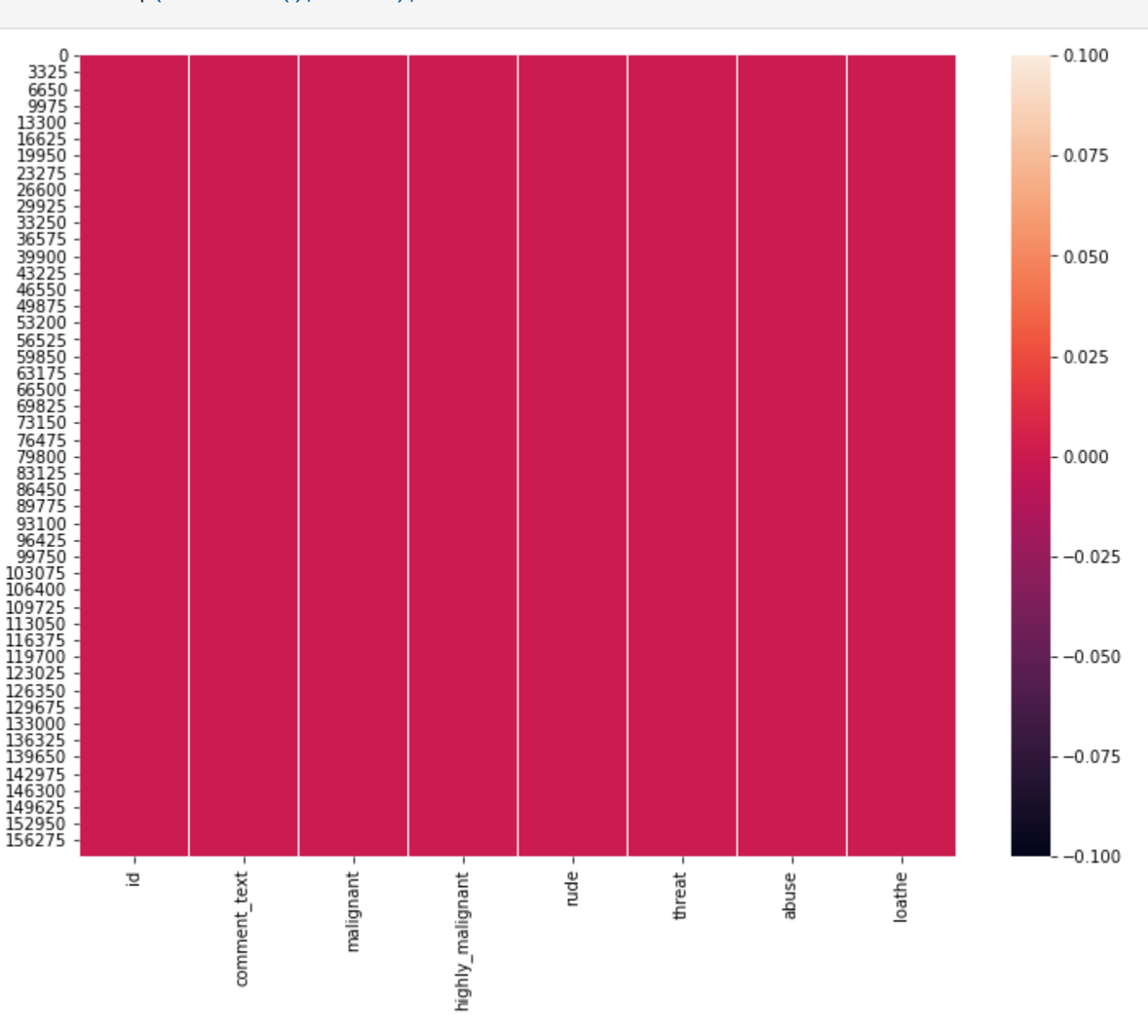
Out[11]:

```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

Checking Isnull Heatmap

In [84]:

```
fig, ax = plt.subplots(figsize=(12,9))
sns.heatmap(df.isnull(), ax=ax);
```



In [12]:

```
df.isnull().sum().sum()
```

Out[12]: 0

Describing Dataset

In [13]:

```
df.describe()
```

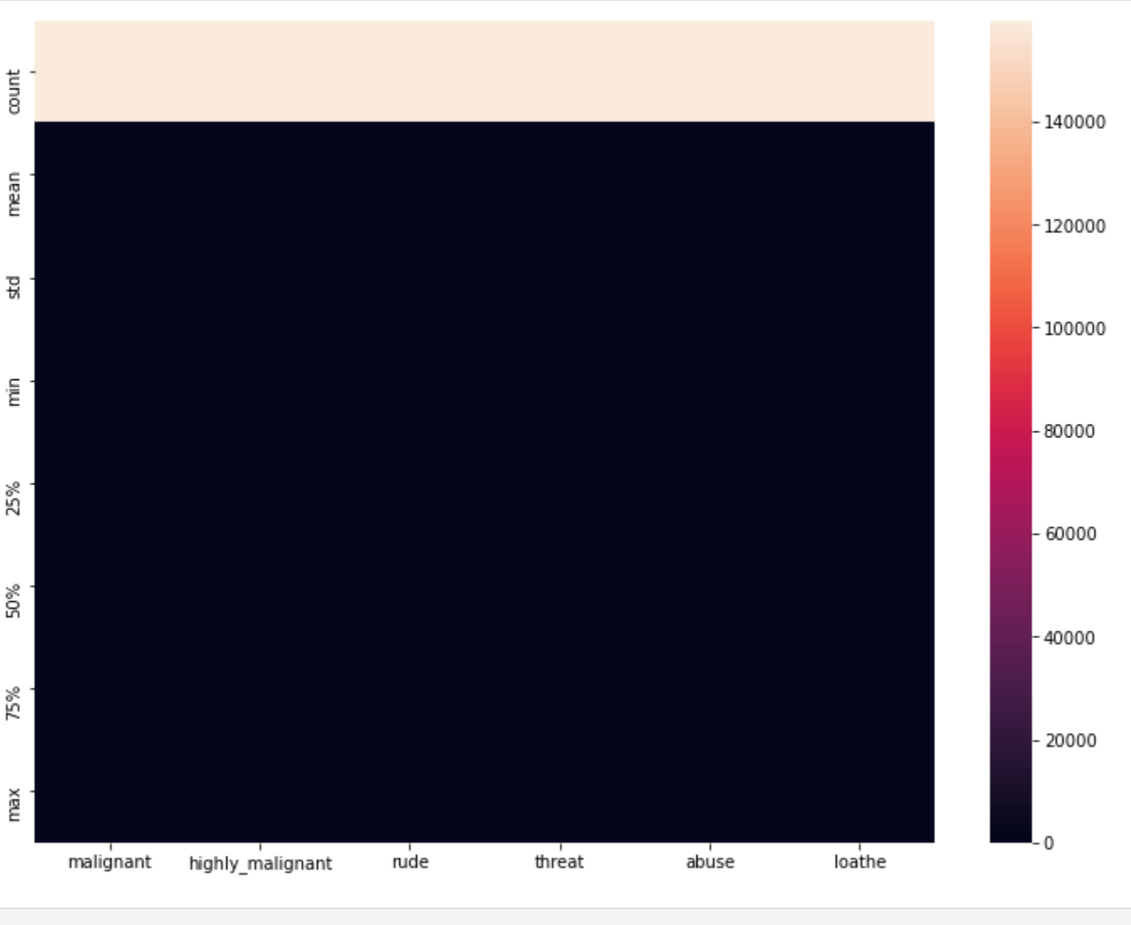
Out[13]:

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Describing Dataset Heatmap

In [17]:

```
import matplotlib.pyplot as plt
import seaborn as sns
fig, ax = plt.subplots(figsize=(12,9))
sns.heatmap(df.describe(), ax=ax);
```



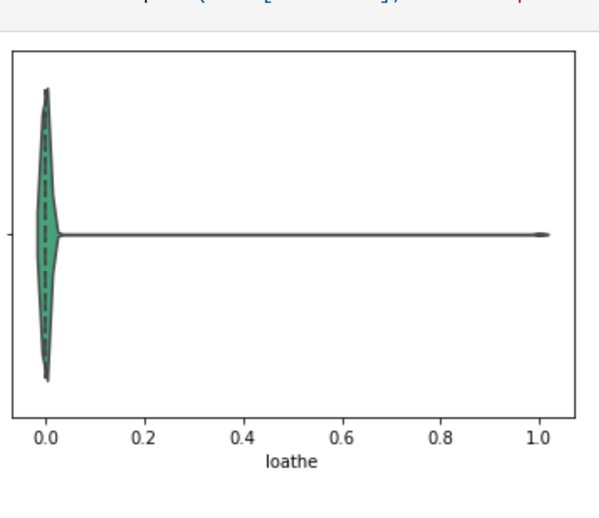
In [18]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 2   malignant         159571 non-null  int64 
 3   highly_malignant  159571 non-null  int64 
 4   rude              159571 non-null  int64 
 5   threat            159571 non-null  int64 
 6   abuse             159571 non-null  int64 
 7   loathe            159571 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

In [20]:

```
sns.violinplot(x=df['loathe'], inner="quartile", color="#36B37E");
```



Correlation

In [83]:

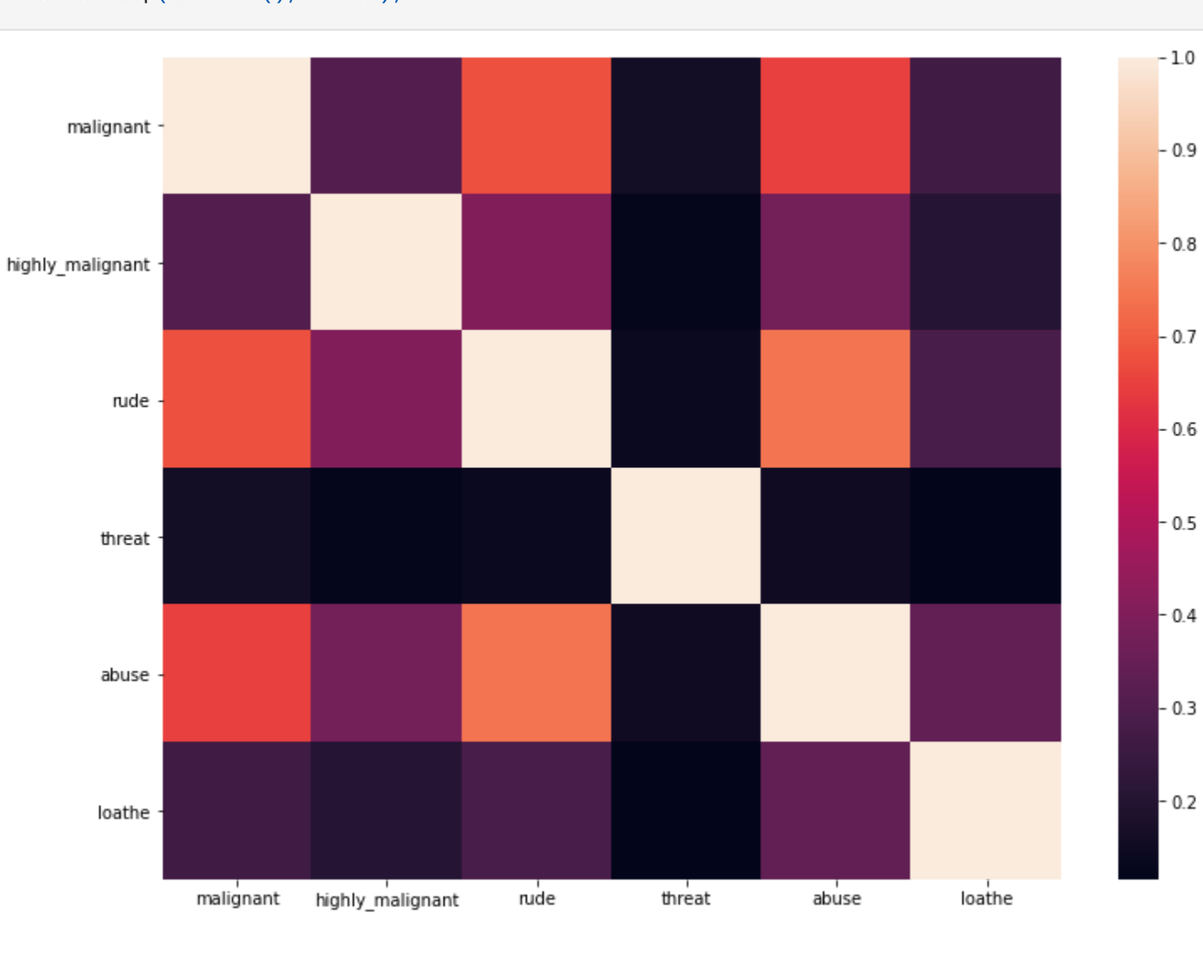
```
df.corr()
```

Out[83]:

	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128
abuse	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736
loathe	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000

In [21]:

```
fig, ax = plt.subplots(figsize=(12,9))
sns.heatmap(df.corr(), ax=ax);
```



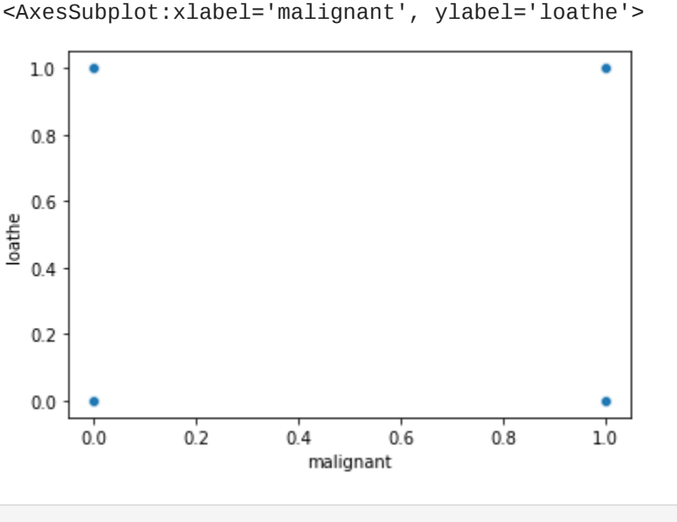
EXPLORATORY DISTRIBUTION ANALYSIS

Scatterplot

In [23]:

```
sns.scatterplot(x="malignant",y="loathe",data=df)
```

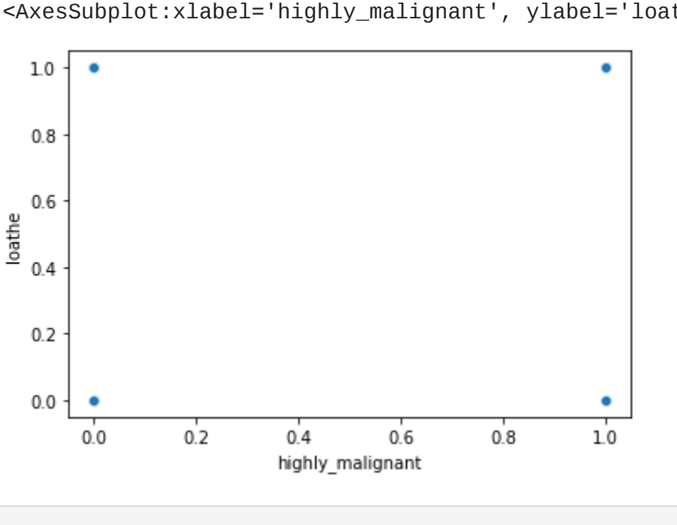
Out[23]: <AxesSubplot:xlabel='malignant', ylabel='loathe'>



In [24]:

```
sns.scatterplot(x="highly_malignant",y="loathe",data=df)
```

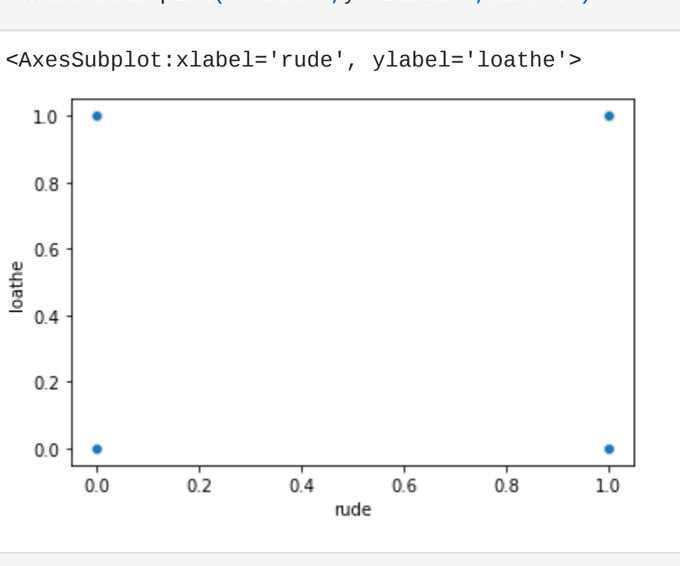
Out[24]: <AxesSubplot:xlabel='highly_malignant', ylabel='loathe'>



In [25]:

```
sns.scatterplot(x="rude",y="loathe",data=df)
```

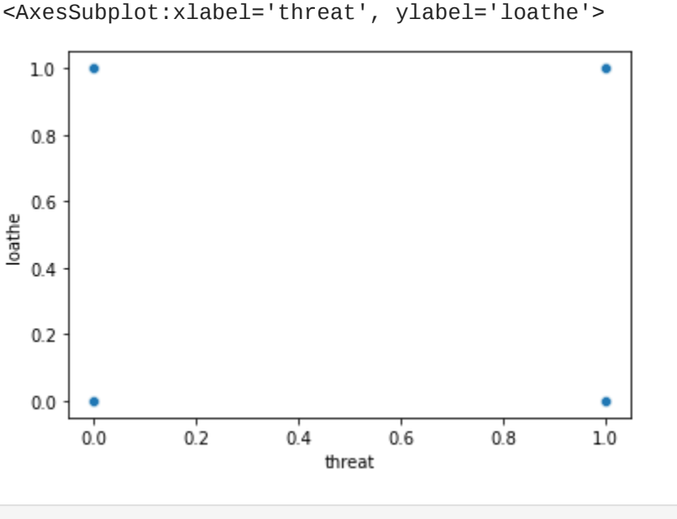
Out[25]: <AxesSubplot:xlabel='rude', ylabel='loathe'>



In [26]:

```
sns.scatterplot(x="threat",y="loathe",data=df)
```

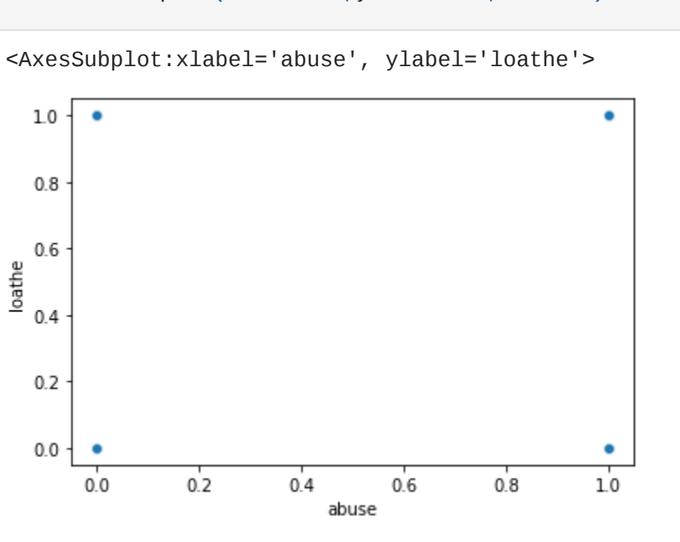
Out[26]: <AxesSubplot:xlabel='threat', ylabel='loathe'>



In [27]:

```
sns.scatterplot(x="abuse",y="loathe",data=df)
```

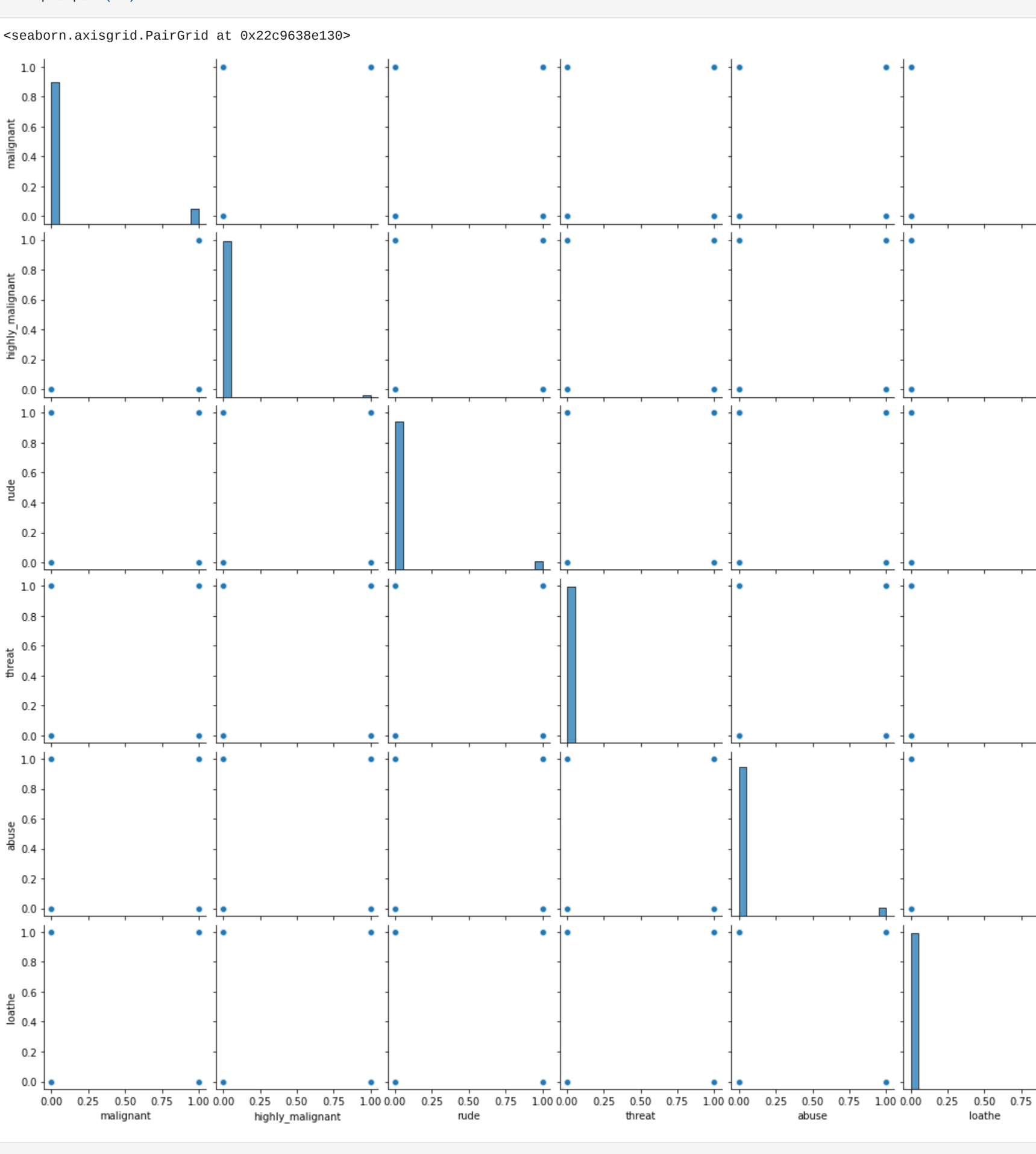
Out[27]: <AxesSubplot:xlabel='abuse', ylabel='loathe'>



In [29]:

```
sns.pairplot(df)
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x22c9638e130>



In [38]:

```
df
```

Out[38]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

skewness

In [37]:

```
df.skew()
```

Out[37]:

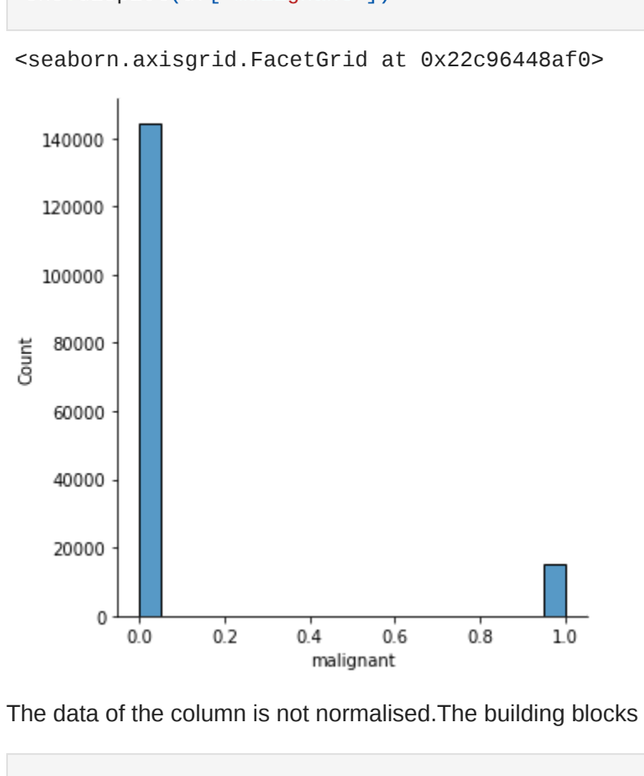
```
malignant            2.745854
highly_malignant     9.851722
rude                 3.992817
threat              18.189001
abuse                4.160540
loathe              10.515923
dtype: float64
```

Normal Distribution Curve

In [39]:

```
sns.displot(df['malignant'])
```

Out[39]: <seaborn.axisgrid.FacetGrid at 0x22c96448af0>

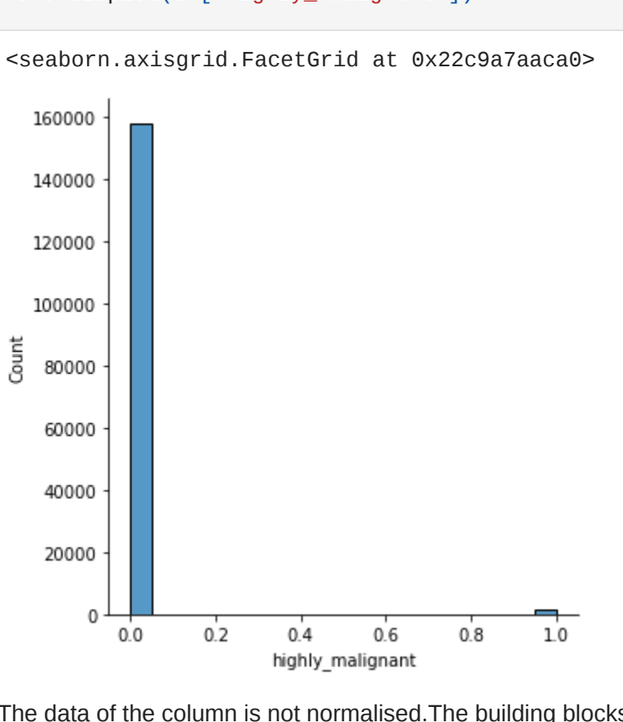


The data of the column is not normalised.The building blocks is out of the normalised curve

In [40]:

```
sns.displot(df['highly_malignant'])
```

Out[40]: <seaborn.axisgrid.FacetGrid at 0x22c9a7aaca0>

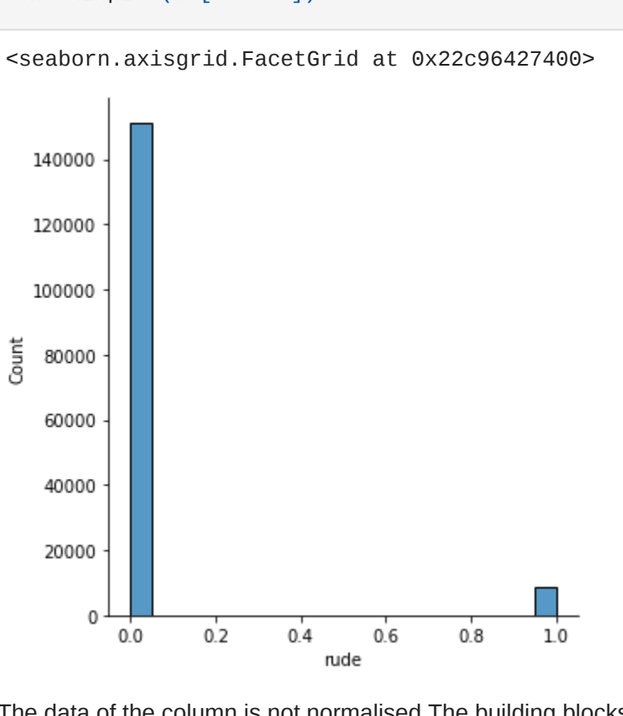


The data of the column is not normalised.The building blocks is out of the normalised curve

In [41]:

```
sns.displot(df['rude'])
```

Out[41]: <seaborn.axisgrid.FacetGrid at 0x22c96427400>

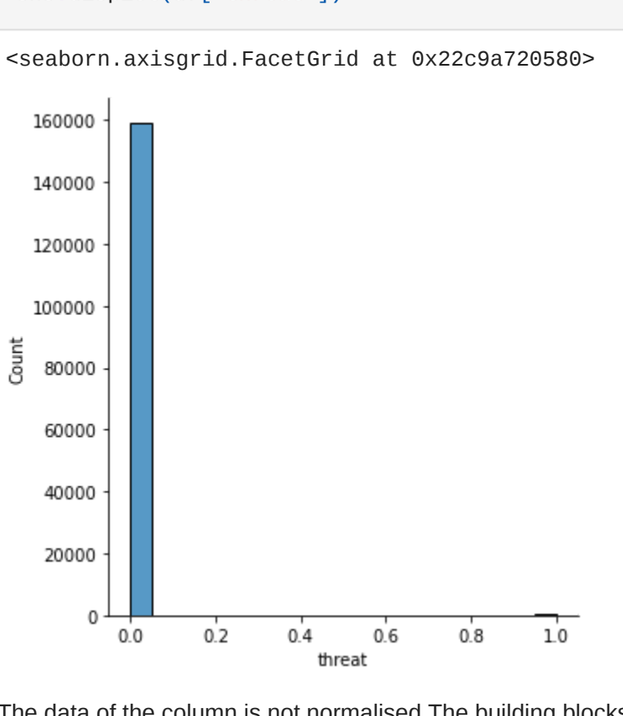


The data of the column is not normalised.The building blocks is out of the normalised curve

In [42]:

```
sns.displot(df['threat'])
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x22c9a720580>

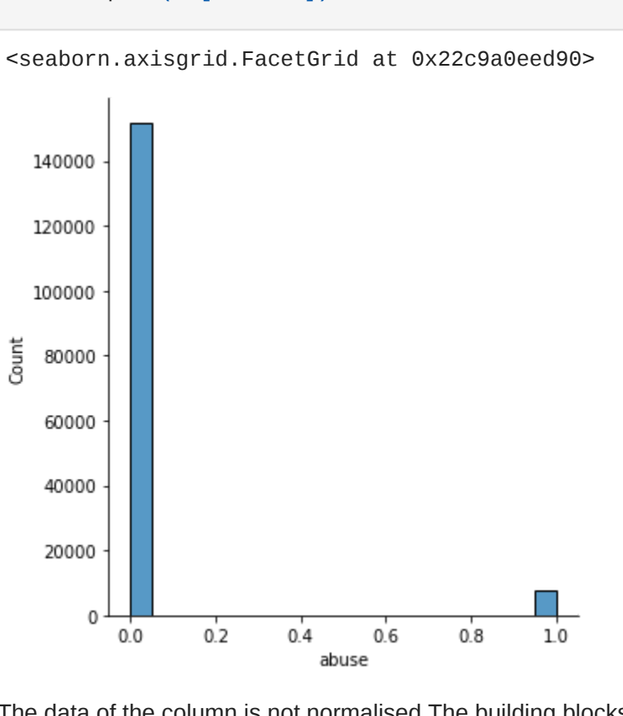


The data of the column is not normalised.The building blocks is out of the normalised curve

In [43]:

```
sns.displot(df['abuse'])
```

Out[43]: <seaborn.axisgrid.FacetGrid at 0x22c9a0eed90>

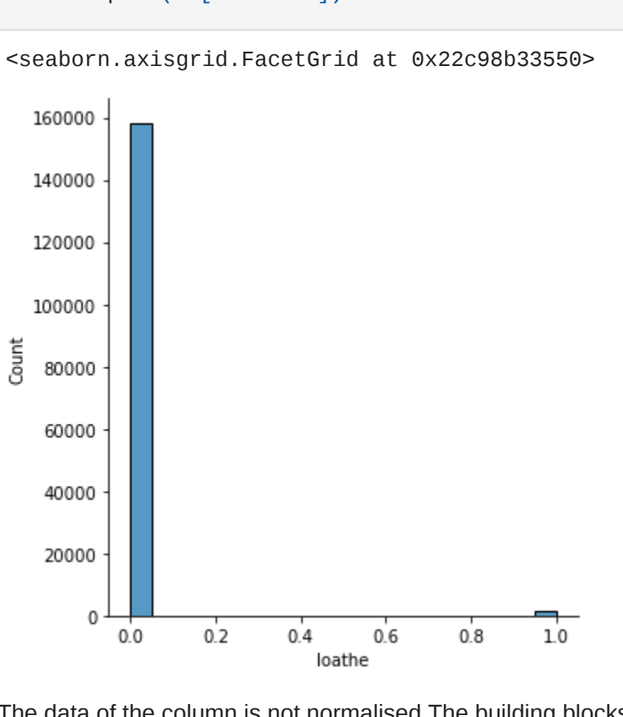


The data of the column is not normalised.The building blocks is out of the normalised curve

In [44]:

```
sns.displot(df['loathe'])
```

Out[44]: <seaborn.axisgrid.FacetGrid at 0x22c98b33550>



The data of the column is not normalised.The building blocks is out of the normalised curve

In [45]:

```
df.corr()['loathe']
```

Out[45]:

```
malignant           0.266009
highly_malignant    0.201600
rude                0.286867
threat              0.115128
abuse               0.337736
loathe              1.000000
Name: loathe, dtype: float64
```

In [52]:

```
from pandas import read_csv
# load dataset
url = 'train.csv'
dataframe = read_csv(url, header=None)
# split into input and output elements
data = dataframe.values
X, y = data[:, :-1], data[:, -1]
print(X.shape, y.shape)
```

(159572, 7) (159572,)

In [53]:

```
X, y = data[:, :-1], data[:, -1]
print(X.shape, y.shape)
```

(159572, 7) (159572,)

In [58]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

(139, 60) (69, 60) (139,) (69,)

In [59]:

```
model = RandomForestClassifier(random_state=1)
model.fit(X_train, y_train)
```

Out[59]: RandomForestClassifier(random_state=1)

In [60]:

```
# make predictions
yhat = model.predict(X_test)
# evaluate predictions
acc = accuracy_score(y_test, yhat)
print('Accuracy: %.3f' % acc)
```

Accuracy: 0.783

In [64]:

```
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# load dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/sonar.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
# split into inputs and outputs
X, y = data[:, :-1], data[:, -1]
print(X.shape, y.shape)
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
# fit the model
model = RandomForestClassifier(random_state=1)
model.fit(X_train, y_train)
# make predictions
yhat = model.predict(X_test)
# evaluate predictions
acc = accuracy_score(y_test, yhat)
print('Accuracy: %.3f' % acc)
```

```
(208, 60) (208,)
(139, 60) (69, 60) (139,) (69,)
Accuracy: 0.783
```

In [56]:

```
from pandas import read_csv
# load dataset
url = 'train.csv'
dataframe = read_csv(url, header=None)
# summarize shape
print(dataframe.shape)
```

(159572, 8)

In [57]:

```
# split into inputs and outputs
X, y = data[:, :-1], data[:, -1]
print(X.shape, y.shape)
```

(208, 60) (208,)

In [71]:

```
df
```

Out[71]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

In [80]:

```
from sklearn.datasets import load_iris
iris = load_iris()

# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted response values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

Gaussian Naive Bayes model accuracy(in %): 95.0

In [81]:

```
# Necessary imports
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Creating the hyperparameter grid
c_space = np.logspace(-5, 8, 15)
param_grid = {'C': c_space}

# Instantiating logistic regression classifier
logreg = LogisticRegression()

# Instantiating the GridSearchCV object
logreg_cv = GridSearchCV(logreg, param_grid, cv = 5)

logreg_cv.fit(X, y)

# Print the tuned parameters and score
print("Tuned Logistic Regression Parameters: {}".format(logreg_cv.best_params_))
print("Best score is {}".format(logreg_cv.best_score_))
```

```
Tuned Logistic Regression Parameters: {'C': 31.622776601683793}
Best score is 0.9800000000000001
```

This is the best models for Malignant-Comments-Classifier.We also suggest that people take into consideration the features that were deemed as most important as seen in the previous section; this might help them estimate the Malignant-Comments-Classifier Prediction.

Thankyou

In [ ]: