

MALIGNANT COMMENTS CLASSIFICATION

Problem Statement

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.

There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

Data Set Description

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.

Highly Malignant: It denotes comments that are highly malignant and hurtful.

Rude: It denotes comments that are very rude and offensive.

Threat: It contains indication of the comments that are giving any threat to someone.

Abuse: It is for comments that are abusive in nature.

Loathe: It describes the comments which are hateful and loathing in nature.

ID: It includes unique Ids associated with each comment text given.

Comment text: This column contains the comments extracted from various social media platforms.

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do good amount of data exploration and derive some interesting features using the comments text column available.

You need to build a model that can differentiate between comments and its categories.

Refer to the data set file provided along with this.

Python library

The “Python library” contains several different kinds of components. It **contains data types that would normally be considered part of the “core” of a language**, such as numbers and lists. ... Some modules are written in C and built in to the Python interpreter; others are written in Python and imported in source form.

```
#import all the library file
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import sklearn
```

```
from sklearn.linear_model import  
LinearRegression
```

```
from sklearn.metrics import accuracy_score
```

```
from sklearn.metrics import  
confusion_matrix,classification_report
```

```
from sklearn.model_selection import  
train_test_split
```

```
from scipy.stats import zscore
```

```
import warnings
warnings.filterwarnings('ignore')
```

Importing data from csv file

#getting the data from train and test

```
import pandas as pd
```

```
df=pd.read_csv('train.csv')
```

```
df1=pd.read_csv('test.csv')
```

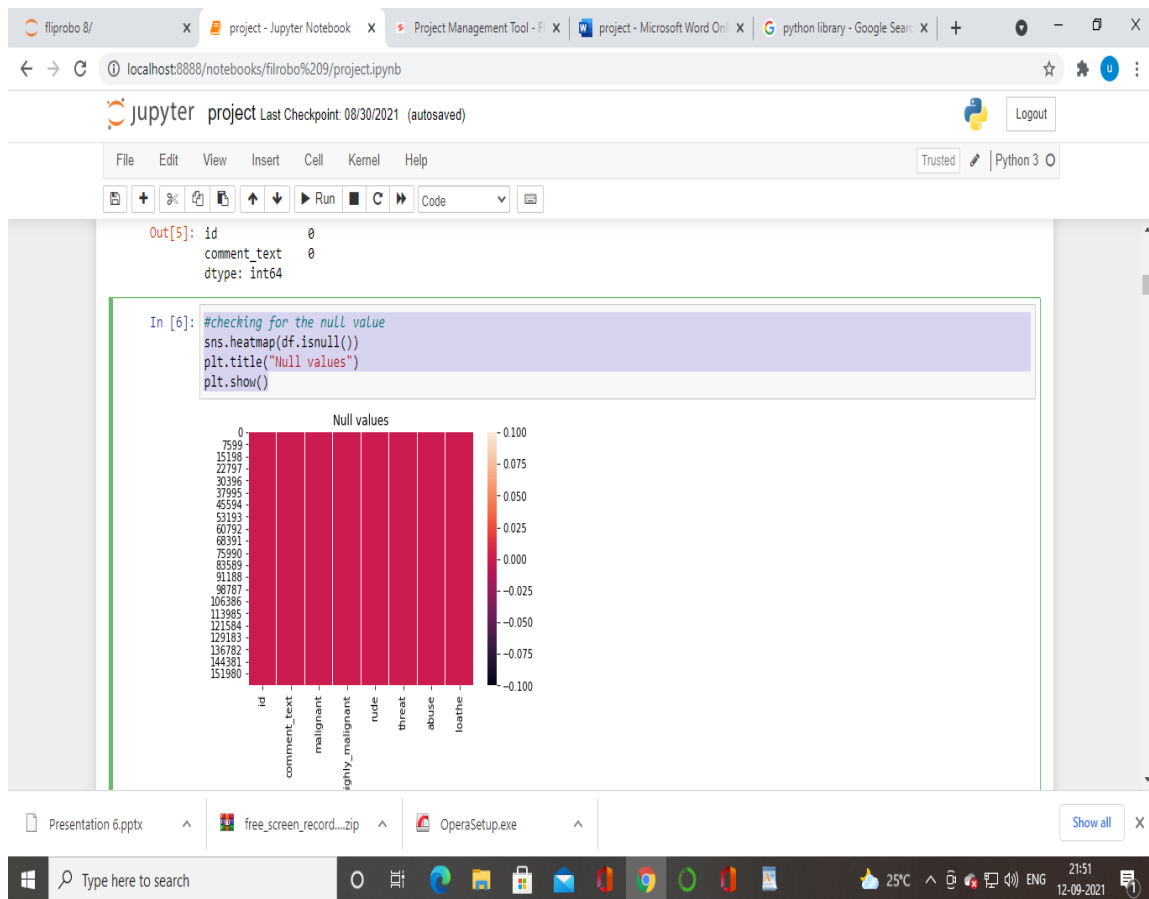
Df1

Checking for the data type

- Whether integer or float

checking for the null value

- `sns.heatmap(df.isnull())`
- `plt.title("Null values")`
- `plt.show()`



#we get mean value and standard deviation value
df.describe()

Browser tabs: filrobo 8/, project - Jupyter Notebook, Project Management Tool - F, project - Microsoft Word Onl, python library - Google Sear, +

Address bar: localhost:8888/notebooks/filrobo%209/project.ipynb

Jupyter interface: project Last Checkpoint: 08/30/2021 (autosaved) | Trusted | Python 3

File Edit View Insert Cell Kernel Help

Run Code

Figure showing 'id' and 'comment_text' with values 138586, 145880 and a color scale from -0.100 to 0.100.

```
In [8]: #we get mean value and standard deviation value
df.describe()
```

Out[8]:

	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

```
In [9]: ##we get mean value and standard deviation value
df1.describe()
```

Out[9]:

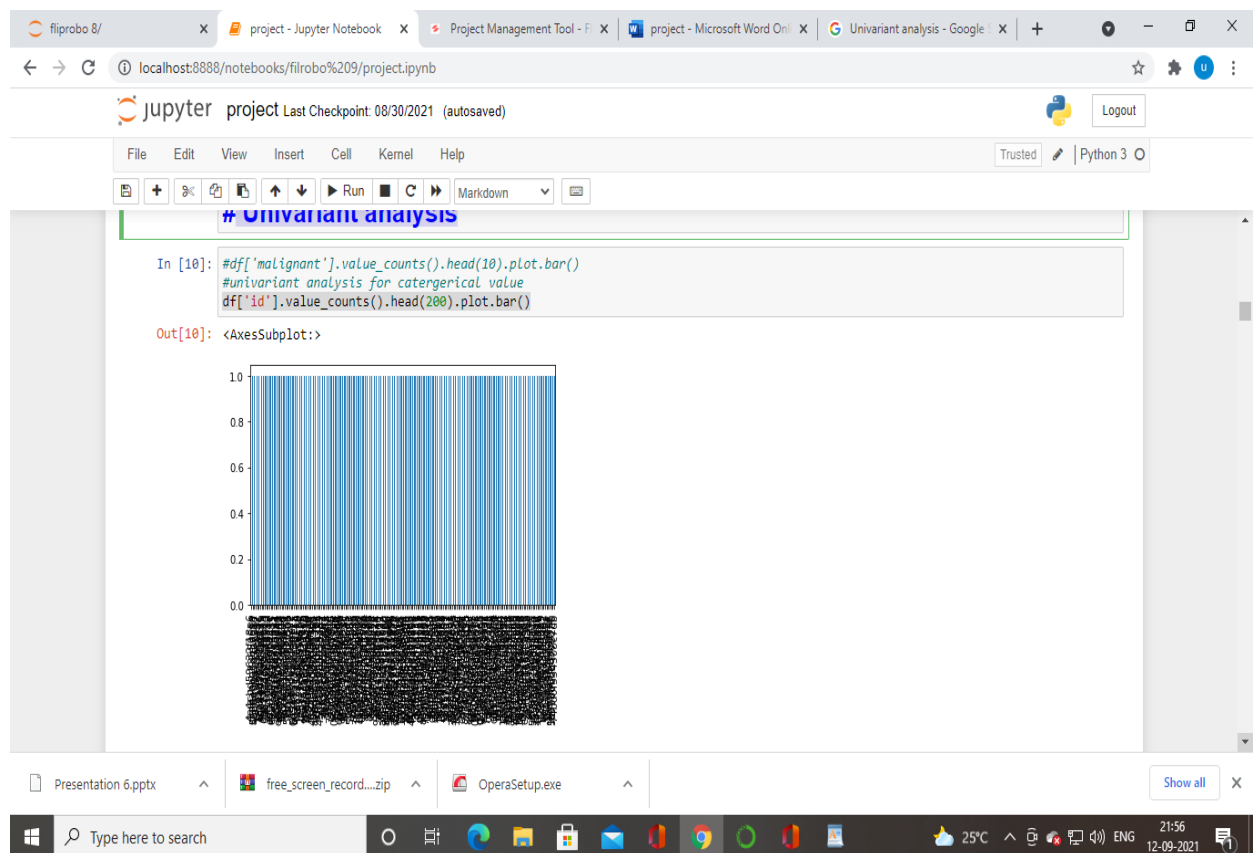
	id	comment_text
count	453464	453464

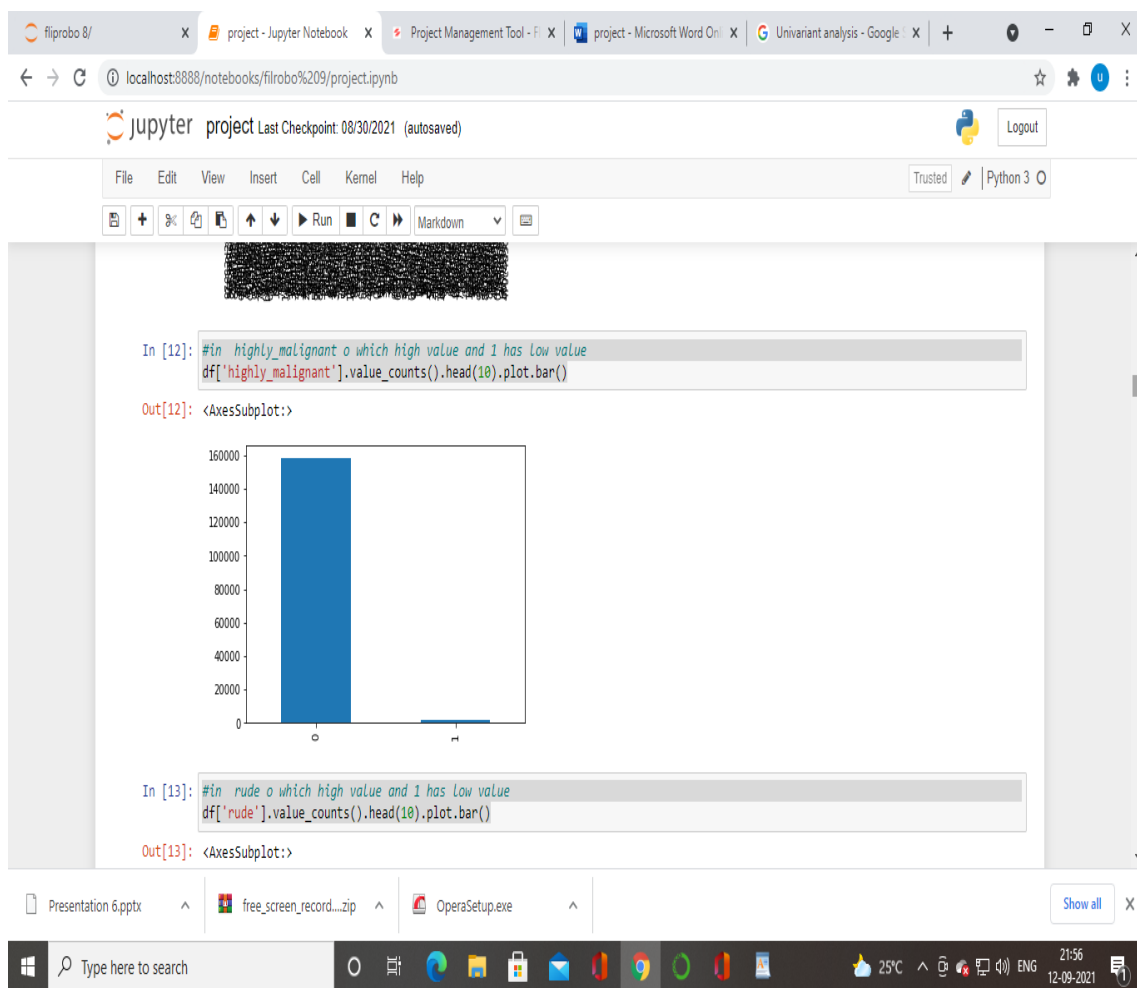
Taskbar: Presentation 6.pptx, free_screen_record..., OperaSetup.exe, Show all

Windows taskbar: Type here to search, 25°C, 21:54, 12-09-2021

Univariate analysis

Univariate analysis is **the simplest form of analyzing data**. “Uni” means “one”, so in other words your data has only one variable. It doesn't deal with causes or relationships (unlike regression) and it's major purpose is to describe; It takes data, summarizes that data and finds patterns in the data.





#getting value of numerical data and categorical

`numeric_data = df.select_dtypes(include=[np.number])`

`categorical_data =`

`df.select_dtypes(exclude=[np.number])`

#numerical data

`numeric_data.columns`

```
Index(['malignant', 'highly_malignant',  
      'rude', 'threat', 'abuse', 'loathe'],  
      dtype='object')
```

```
#counting the value data from train.csv
```

```
df.stack().value_counts()
```

```
df.corr()
```

When the data contains only one variable and doesn't deal with a causes or effect relationships then a Univariate analysis technique is used. ...

The key objective of Univariate analysis is to simply describe the data to find patterns **within** the data.

The screenshot shows a Jupyter Notebook running in a web browser. The notebook is titled 'project' and has a last checkpoint of 08/30/2021. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running cells, and code execution. The notebook content shows two code cells. The first cell, labeled 'In [28]:', contains the code `df1.corr()`. The second cell, labeled 'In [29]:', contains the code `df.corr()`. The output of the second cell is a correlation matrix for six variables: malignant, highly_malignant, rude, threat, abuse, and loathe. The matrix is a 6x6 table with values ranging from 0.000000 to 1.000000. Below the matrix, there is a text box containing the following text: 'rude and malignant are highly correlated', 'abuse and malignant are highlt correlated', and 'abuse and rude are highly correlated'. The notebook also shows a third code cell, labeled 'In [30]:', containing the code `sns.pairplot(df)`. The bottom of the screenshot shows the Windows taskbar with various application icons and the system clock displaying 22:03 on 12-09-2021.

```
In [28]: df1.corr()
```

```
Out[28]:
```

```
In [29]: df.corr()
```

```
Out[29]:
```

	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

rude and malignant are highly correlated
abuse and malignant are highlt correlated
abuse and rude are highly correlated

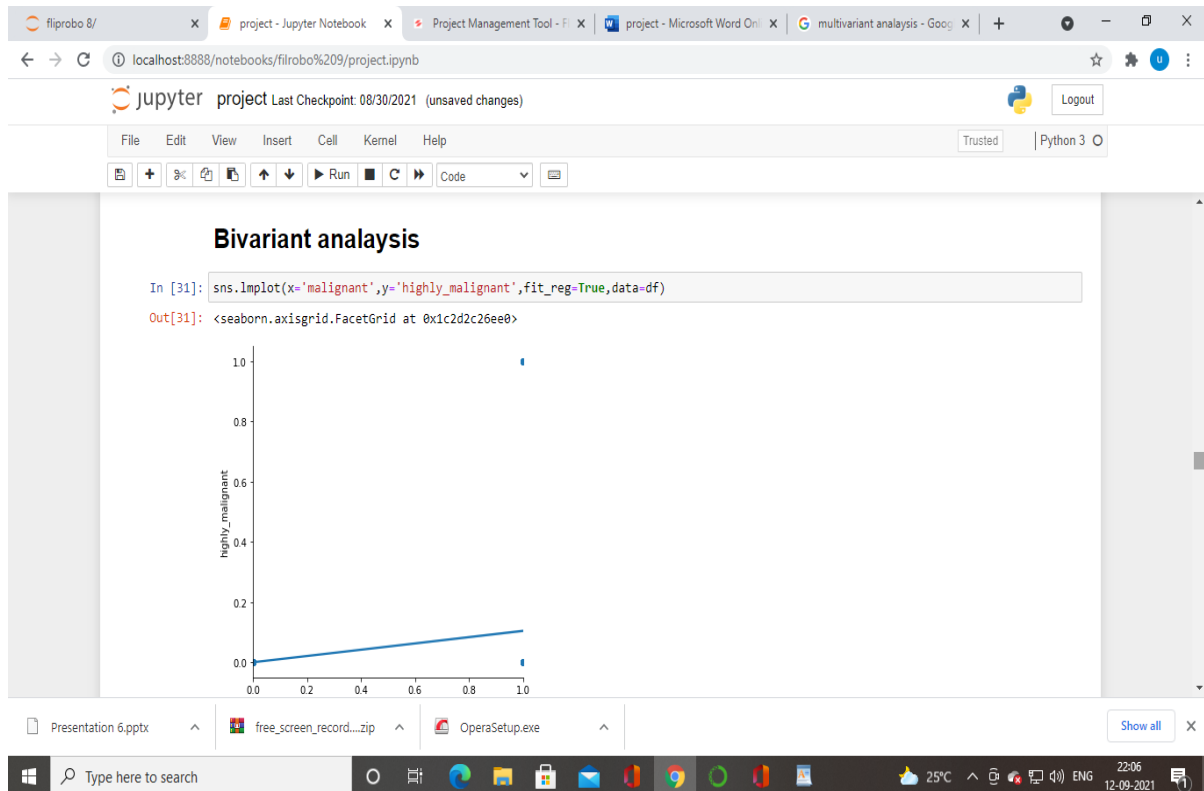
multivariant analysis

```
In [30]: sns.pairplot(df)
```

multivariant analysis

Multivariate analysis is a **set of techniques used for analysis of data sets that contain more than one variable**, and the techniques are especially valuable when working with correlated variables.

Bivariant analysis



Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form.

#converting categorical value to numerical value

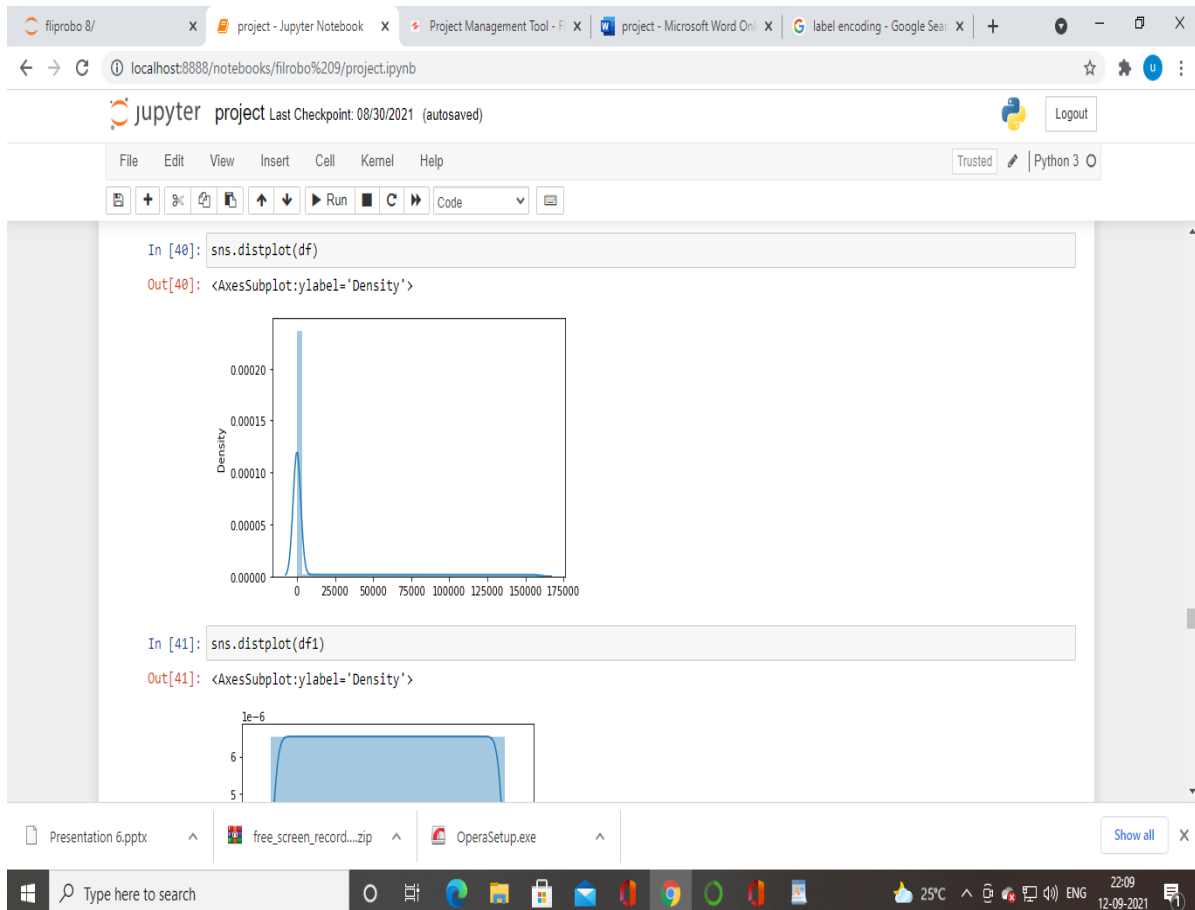
```
from sklearn.preprocessing import LabelEncoder
```

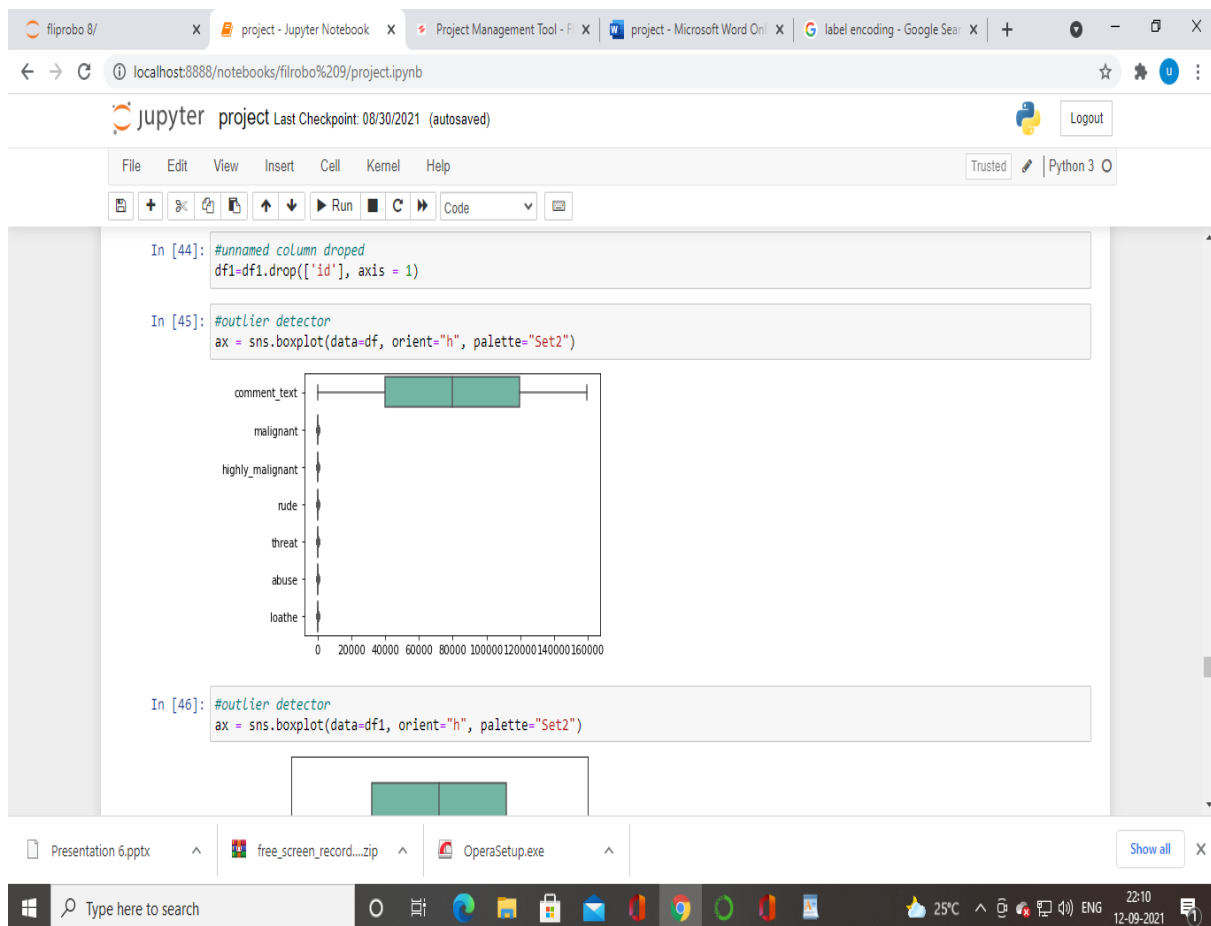
```
le=LabelEncoder()
```

```
list1=['id', 'comment_text']
```

for val in list1:

df[val]=le.fit_transform(df[val].astype(str))





removing outliers

from scipy.stats import zscore

z_score=abs(zscore(df))

print(df.shape)

df=df.loc[(z_score<3).all(axis=1)]

print(df.shape)

StandardScaler uses to reduce the rows

```
from sklearn.preprocessing import StandardScaler
import numpy as np
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
print(df)
```

```
1.from sklearn.linear_model import
   LogisticRegression
2.from sklearn.neighbors import
   KNeighborsClassifier
3.from sklearn.naive_bayes import GaussianNB
4.from sklearn.tree import
   DecisionTreeClassifier
5.from sklearn.svm import SVC
6.from sklearn.ensemble import
   RandomForestClassifier
```

- Seperating independent variable

```
X=df.drop('loathe',axis=1)
y=df['loathe']
```

The screenshot shows a Jupyter Notebook running in a web browser. The notebook contains two code cells. The first cell imports several classifiers from the sklearn library. The second cell, labeled 'In [56]:', defines a RandomForestClassifier, fits it to training data, and prints its performance metrics. The output shows a perfect score of 1.0 across all metrics.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

```
In [56]: rf=RandomForestClassifier()
rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

1.0
[[31537]]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	31537
accuracy			1.00	31537
macro avg	1.00	1.00	1.00	31537
weighted avg	1.00	1.00	1.00	31537

```
In [ ]:
```

```
In [57]: knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
```

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model.

[1. 1. 1. 1. 1.]1.0

Cross-validation is a **method for robustly estimating test-set performance (generalization)** of a model. Grid-search is a way to select the best of a family of models, parametrized by a grid of parameters

```
GCV_pred=GCV.best_estimator_.predict(x_test)
```

```
accuracy_score(y_test,GCV_pred)
```

1.0