MALIGNANT COMMENTS PREDICTION

Submitted by:

SAYAN KUMAR BHUNIA

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped me and guided me in completion of the project.

- https://towardsdatascience.com/

INTRODUCTION

➢ Business Problem Framing

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.

➢ Conceptual Background of the Domain Problem

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.

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but “u are an idiot” is clearly offensive.

➢ Motivation for the Problem Undertaken

Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

Analytical Problem Framing

➢ Mathematical/ Analytical Modelling of the Problem

We shall build a supervised Classification model to predict the price of a car based on given features.

Now, when we talk about building a supervised Classification catering to certain use-case, following three things come into our minds:

- **Data** appropriate to the business requirement or use-case we are trying to solve
- A **Classification model** which we think (or, rather assess) to be the best for our solution.
- **Optimize** the chosen model to ensure best performance.

➢ Data Sources and their formats

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.

```
2]: train_df=pd.read_csv(r"C:\Users\sayan\OneDrive\Desktop\fliprobo projects\Malignant Comments Classifier Project\train.csv")
    train_df
```

2]:

	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

➢ Data Pre-processing

Following steps have been performed on the data.

- **checking missing values**-
  - If there is any missing value present in your data set then for a better and correct accuracy you have to impute it.
  - If missing data present in object type column, then you have to take most frequent value for your missing data.
  - If missing data present in int or float type column then use mean/median for missing value.

In the following case no missing value present:

```
[150]: train_df.isnull().sum()

[150]: comment_text        0
       malignant           0
       highly_malignant    0
       rude                0
       threat              0
       abuse               0
       loathe              0
       dtype: int64
```

- **Stemming**- Stemming is a natural language processing technique that lowers inflection in words to their root forms, hence aiding in the preprocessing of text, words, and documents for text normalization.
  PorterStemmer() is a module in NLTK that implements the Porter Stemming technique.

```
In [167]: port_stem=PorterStemmer()

In [168]: def stemming(comment_text):
              stemmed_comment_text = re.sub('[^a-zA-z]',' ',comment_text)##will re
              stemmed_comment_text = stemmed_comment_text.lower() ##converting all
              stemmed_comment_text = stemmed_comment_text.split()
              #stemmed_comment_text = [port_stem(word) for word in stemmed_comment
              stemmed_comment_text = ' '.join(stemmed_comment_text)
              return stemmed_comment_text
```

- **TfidfVectorizer-**. Tf-idf stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

- **Feature scaling-** Feature Scaling ensures that all features will get equal importance in supervised classifier models. Standard scaler was used to scale all features in the data.

- **Reducing dimension of the data-** Sklearn's pca can be used to apply principal component analysis on the data. This helped in finding the vectors of maximal variance in the data.
  In this case iam not using it.

- **Outliers detection-** In simple words, an outlier is an observation that diverges from an overall pattern on a sample.

  In the following case no need to detect outliers.

➢ Software Requirements and library Used

```
: import pandas
  import pandas as pd
  import numpy as np
  import seaborn as sns
  import matplotlib.pyplot as plt
  import re
  from nltk.corpus import stopwords
  from nltk.stem.porter import PorterStemmer
  from sklearn.feature_extraction.text import TfidfVectorizer
  import warnings
  warnings.filterwarnings('ignore')
```

```
: !pip install nltk

  Requirement already satisfied: nltk in c:\users\sayan\anaconda3\lib\site-packages (3.6.5)
  Requirement already satisfied: click in c:\users\sayan\anaconda3\lib\site-packages (from nltk) (8.0.3)
  Requirement already satisfied: joblib in c:\users\sayan\anaconda3\lib\site-packages (from nltk) (1.1.0)
  Requirement already satisfied: regex>=2021.8.3 in c:\users\sayan\anaconda3\lib\site-packages (from nltk
  Requirement already satisfied: tqdm in c:\users\sayan\anaconda3\lib\site-packages (from nltk) (4.62.3)
  Requirement already satisfied: colorama in c:\users\sayan\anaconda3\lib\site-packages (from click->nltk
```

```
: import nltk
  nltk.download()

  showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml

: True
```

- **NumPy**
  NumPy is a popular Python library for multi-dimensional array and matrix processing because it can be used to perform a great variety of mathematical operations. Its capability to handle linear algebra, Fourier transform, and more, makes NumPy ideal for machine learning and artificial intelligence (AI) projects, allowing users to manipulate the matrix to easily improve machine learning performance. NumPy is faster and easier to use than most other Python libraries.

- **Scikit-learn**

Scikit-learn is a very popular machine learning library that is built on NumPy and SciPy. It supports most of the classic supervised and unsupervised learning algorithms, and it can also be used for data mining, modelling, and analysis.

- **Seaborn**
  Seaborn is another open-source Python library, one that is based on Matplotlib (which focuses on plotting and data visualization) but features Pandas' data structures. Seaborn is often used in ML projects because it can generate plots of learning data. Of all the Python libraries, it produces the most aesthetically pleasing graphs and plots, making it an effective choice if you'll also use it for marketing and data analysis.

- **Pandas**
  Pandas is another Python library that is built on top of NumPy, responsible for preparing high-level data sets for machine learning and training. It relies on two types of data structures, one-dimensional (series) and two-dimensional (Data Frame). This allows Pandas to be applicable in a variety of industries including finance, engineering, and statistics. Unlike the slow-moving animals themselves, the Pandas library is quick, compliant, and flexible.

➢ Class imbalance problem

The first challenge we hit upon exploring the data, is class imbalanced problem. Imbalance data will lead to a bad accuracy of a model. To achieve better accuracy, we'll balance the data by using Smote Over Sampling or under sampling Method .But in this project data is balanced so I am not using it.

Model/s Development and Evaluation

➢ Run and evaluate selected models

Let’s select our classification model for this project:

- Decission Tree Classifier
- RandomForestClassifier

➢ Testing of Identified Approaches (Algorithms)

```
]: import sklearn
   from sklearn.ensemble import RandomForestClassifier
   from sklearn.tree import DecisionTreeClassifier
   from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
   from sklearn.model_selection import train_test_split
```

```
]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=30)
```

```
]: rf=RandomForestClassifier(n_estimators=100,random_state=50)
   dt = DecisionTreeClassifier()
   model=[rf,dt]
   for m in model:
       m.fit(x_train,y_train)
       predm=m.predict(x_train)
       print('Training accuracy is {}'.format(accuracy_score(y_train,predm)))
       predm=m.predict(x_test)
       print('accuracy_score of',m ,'is:')
       print(accuracy_score(y_test,predm))
       print(confusion_matrix(y_test,predm))
       print(classification_report(y_test,predm))
```

```
Training accuracy is 0.9988988263099938
accuracy_score of RandomForestClassifier(random_state=50) is:
0.9560494652406417
[[42317   598]
 [ 1506  3451]]
```

➢ Key Metrics for success in solving problem under consideration

Selection of a model requires evaluation and evaluation requires a good metric. This is indeed important. If we optimize a model based on incorrect metric, then, our model might not be suitable for the business goals.

We have a number of metrics, for example, accuracy, recall, precision, F1 score, area under receiver operating characteristic curve, to choose from.

CONCLUSION

Key aspects of building successful classifier are:

- Selecting correct data according to the purpose or problem statement.
- Proper processing and understanding of the data
- Selecting the model and optimizing the model.

I have used tfidfvectorizer for convert object data type into int datatype as machine doesn't understand object type data.

I have used three Classification model and found Random Forest Classifier to be the best fit. It is giving 99% accuracy.