FLIP ROBO

MALIGNANT COMMENTS CLASSIFIER

Submitted by:

Komal Vijay Ghatvilkar

ACKNOWLEDGMENT

The success & outcome of this project were possible by the guidance and support from FlipRobo.

It was not possible to done without research from different machine learning sites on Google.

I referred DataTrained material for more information that helped me completion of the project.

Thank you……..!!!

It has become evident that human behaviour is changing; our emotions are getting attached to the likes, comments and tags we receive on social media. We get both good and bad comments but seeing hateful words, slurs and harmful ideas on digital platforms on a daily basis make it look normal when it shouldn't be. The impact of malignant comments is much more catastrophic than we think. It not only hurts one's self-esteem or deters people from having meaningful discussions, but also provokes people to such sinister acts.

With the proliferation of smart devices and mobile and social network environments, the social side effects of these technologies, including cyberbullying through malicious comments and rumors, have become more serious. Malicious online comments have emerged as an unwelcome social issue worldwide. In the U.S., a 12-year-old girl committed suicide after being targeted for cyberbullying in 2013. In Singapore, 59.4% of students underwent at least some kind of cyberbullying, and 28.5% were the targets of nasty online comments in 2013. In Australia, Charlotte Dawson, who at one time hosted the "Next Top Model" TV program, committed suicide in 2012 after being targeted with malicious online comments. In Korea, where damage caused by malicious comments is severe, more than 20% of Internet users, from teenagers to adults in their 50s, posted malicious comments in 2011.

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 will be tagged as unoffensive, but "u are an idiot" is clearly offensive. 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.

We have done the following analysis of the dataset where we Imported necessary libraries so that we can work on datasets with the Jupyter notebook.

- `import numpy as np`
- `import pandas as pd`
- `import matplotlib.pyplot as plt`
- `import seaborn as sns`
- `import warnings`

- `warnings.filterwarnings('ignore')`

Data contains two files Train and Test which has 159571 entries each having 8 variables and 153164 entries each having 2 variables accordingly.

After reading the dataset I proceed with the EDA.

- `df = pd.read_csv(r'C:\Users\HP\Desktop\train.csv')`
- `df1 = pd.read_csv(r'C:\Users\HP\Desktop\test.csv')`

I checked the description of Training data with .info() method.

- `df.info()`
- `df1.info()`

After .describe() done found the statistical description of data & found no null values so performed the task further.

- `df.describe()`
- `df1.describe()`
- `df.isnull().sum()`

With the correlation among all the columns checked the correlation and most of the data is positively correlated with each other.

- `df.corr()`

In data visualization done the following visualizations :

First used Correlation Matrix for showing the correlation between all columns with Heatmap.

From the output of correlation matrix, we can see that it is symmetrical i.e. the bottom left is same as the top right and positively correlated.

- `corr_mat=df.corr()`
- `# Size of the canvas`
- `plt.figure(figsize=[10,10])`
- `#Plot Correlation Matrix`
- `sns.heatmap(corr_mat,annot=True) # annot represnts each value encoded in heatmap`
- `plt.title('Correlation Matrix')`
- `plt.show()`

The result of the Correlation Matrix is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/Correlation%20Matrix.png

Second, used Histogram for all dataset for visualizing the data individually.

The visualization plot shows that each variable distributed differently and as we can see data has categorical values, so used histogram for better understanding to show the distribution.

```
- df.hist(bins=20,figsize=(10,10))
- plt.show()
```

The result of Histogram is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/Histogram.png

Then I checked the skewness of the data. After that checked the outliers if any, found very less so didn't removed the outliers. I used boxplot to check the outliers in dataset.

The result of the Boxplot is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/BoxPlot%20Outliers.png

Then to show the bad words or separate toxic words and comments did the following task by using NLTK libraries and adding column for checking the final length of the comments :-

```
- from nltk.stem import WordNetLemmatizer
- import nltk
- from nltk.corpus import stopwords
- import string

- df['length'] = df['comment_text'].str.len()
- df.head()

- # Convert all messages to lower case
- df['comment_text'] = df['comment_text'].str.lower()

- # Replace email addresses with 'email'
- df['comment_text']     =     df['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
-                                  'emailaddress')

- # Replace URLs with 'webaddress'
- df['comment_text']   =   df['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
-                                   'webaddress')

- # Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
- df['comment_text'] = df['comment_text'].str.replace(r'£|\$', 'dollers')
-
- # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
- df['comment_text']    =    df['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
-                                   'phonenumber')

-
```

```
-   # Replace numbers with 'numbr'
-   df['comment_text'] = df['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')


-   df['comment_text'] = df['comment_text'].apply(lambda x: ' '.join(
-       term for term in x.split() if term not in string.punctuation))

-   stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im',
    'dont', 'doin', 'ure'])
-   df['comment_text'] = df['comment_text'].apply(lambda x: ' '.join(
-       term for term in x.split() if term not in stop_words))

-   lem=WordNetLemmatizer()
-   df['comment_text'] = df['comment_text'].apply(lambda x: ' '.join(
-    lem.lemmatize(t) for t in x.split()))

-   df['clean_length'] = df.comment_text.str.len()
-   df.head()

-    df.length.sum()
-   df.clean_length.sum()

-   import sys
-   print(sys.executable)
```

Then plotted graph to see the offensive words or bad words exists :-

```
-   #Getting sense of loud words which are offensive
-   from wordcloud import WordCloud
-   hams = df['comment_text'][df['malignant']==1]
-   spam_cloud                                                             =
    WordCloud(width=600,height=400,background_color='black',max_words=50).generat
    e(' '.join(hams))
-   plt.figure(figsize=(10,8),facecolor='k')
-   plt.imshow(spam_cloud)
-   plt.axis('off')
-   plt.tight_layout(pad=0)
-   plt.show()
```

The result of the Wordcloud is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/WordCloud.png

Imported some libraries for model building and plotted a pie chart of distribution of comments as per the categories given so that we can find out how much percentage of data is distributed in each category :-

```
-   from sklearn.naive_bayes import MultinomialNB
-   from sklearn.model_selection import train_test_split
-   from      sklearn.metrics      import      accuracy_score,      confusion_matrix,
    classification_report,roc_curve,roc_auc_score,auc
-   from sklearn.model_selection import train_test_split
```

```
- from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
- from sklearn.linear_model import LogisticRegression
- from sklearn.model_selection import cross_val_score,GridSearchCV
- from sklearn.naive_bayes import MultinomialNB
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
- from sklearn.naive_bayes import GaussianNB
- from sklearn.linear_model import LogisticRegression
- from sklearn.svm import SVC
- from sklearn.tree import DecisionTreeClassifier
```

```
- cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
- df_distribution = df[cols_target].sum()\
-                             .to_frame()\
-                             .rename(columns={0: 'count'})\
-                             .sort_values('count')

- df_distribution.plot.pie(y='count',
-                                       title='Label distribution over comments',
-                                       figsize=(5, 5))\
-                             .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

The result of the Pie Chart of columns distribution is on following GitHub link.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/Comments%20Distribution%20Graph.png

```
- target_data = df[cols_target]

- df['bad'] =df[cols_target].sum(axis =1)
- print(df['bad'].value_counts())
- df['bad'] = df['bad'] > 0
- df['bad'] = df['bad'].astype(int)
- print(df['bad'].value_counts())

- # Convert text into vectors using TF-IDF
- from sklearn.feature_extraction.text import TfidfVectorizer
- tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
- features = tf_vec.fit_transform(df['comment_text'])
- x = features

- df.shape
- df1.shape
```

And then did the Train Test Split :-

```
- y=df['bad']
```

- x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.30)
- y_train.shape,y_test.shape

With the best suitable model building for categorical dataset performed The task and used various model bulding techniques like Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Xgboost Classifier, AdaBoost Classifier & Kneighbors Classifier.

```
- # LogisticRegression
- LG = LogisticRegression(C=1, max_iter = 3000)
- LG.fit(x_train, y_train)
- y_pred_train = LG.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = LG.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- # DecisionTreeClassifier
- DT = DecisionTreeClassifier()
- DT.fit(x_train, y_train)
- y_pred_train = DT.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = DT.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- #RandomForestClassifier
- RF = RandomForestClassifier()
- RF.fit(x_train, y_train)
- y_pred_train = RF.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = RF.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- # xgboost
- import xgboost
- xgb = xgboost.XGBClassifier()
- xgb.fit(x_train, y_train)
- y_pred_train = xgb.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = xgb.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- #AdaBoostClassifier
- ada=AdaBoostClassifier(n_estimators=100)
- ada.fit(x_train, y_train)
- y_pred_train = ada.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = ada.predict(x_test)
```

- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

- #KNeighborsClassifier
- knn=KNeighborsClassifier(n_neighbors=9)
- knn.fit(x_train, y_train)
- y_pred_train = knn.predict(x_train)
- print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
- y_pred_test = knn.predict(x_test)
- print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
- print(confusion_matrix(y_test,y_pred_test))
- print(classification_report(y_test,y_pred_test))

Next did Cross Validation with "cross_val_score" for the models used & it shows the output :-

- from sklearn.model_selection import cross_val_score
- scr=cross_val_score(LG,x,y,cv=5)
- print("Cross Validation Score Of Logistic Regression Model :",scr.mean())
- scr1=cross_val_score(DT,x,y,cv=5)
- print("Cross Validation Score Of Decision Tree Model :",scr1.mean())
- scr2=cross_val_score(RF,x,y,cv=5)
- print("Cross Validation Score Of Random Forest Model :",scr2.mean())
- scr3=cross_val_score(ada,x,y,cv=5)
- print("Cross Validation Score Of Adaboost Model :",scr3.mean())
- scr4=cross_val_score(knn,x,y,cv=5)
- print("Cross Validation Score Of KNeighbors Model :",scr4.mean())
- scr5=cross_val_score(xgb,x,y,cv=5)
- print("Cross Validation Score Of xgboost Model :",scr5.mean())

Then checked the regularization with GridSearchCV to perform regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

- from sklearn.model_selection import GridSearchCV
- # Create parameters list to pass in GridSearchCV
- parameters={'max_features':['auto','sqrt','log2'],'max_depth':[4,5,6,7,8],'criterion':['gini','entropy']}
- GCV=GridSearchCV(RandomForestClassifier(),parameters,cv=5,scoring="accuracy")
- GCV.fit(x_train,y_train) # fitting the data in model
- GCV.best_params_ # Printing the best parameter found by GridSearchCV
- GCV_pred=GCV.best_estimator_.predict(x_test) # predicting with best parameters
- accuracy_score(y_test,GCV_pred) # checking final accuracy

With ROC AUC Curve we achieved best accuracy of dataset.

- from sklearn.metrics import plot_roc_curve
- plot_roc_curve(GCV.best_estimator_,x_test,y_test)
- plt.title("ROC AUC Plot")

- plt.show()
- # Conclusion:
- Attrition=np.array(y_test)
- Predicted=np.array(LG.predict(x_test))
- df_1=pd.DataFrame({'original':Attrition,'predicted':Predicted},index=range(len(Attrition)))
- # saving the dataframe
- df_1.to_csv('Conclusion.csv')
- # Model Saving
- import pickle
- filename = 'MalignantComments.pkl'
- pickle.dump(RF,open(filename,'wb'))

Finally came to the Conclusion as per the results found those are the best model is Random Forest Classifier showing the result for the dataset with approx. 96% accuracy and as per the AUC score it's 94% which shows the data is correct and accurate to proceed.

Please find the GitHub links for ROC AUC Curve to refer.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/ROC%20AUC%20Plot.png

Please find the GitHub link for conclusion data file to refer.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/Conclusion.csv

Please find the GitHub link for Jupyter Notebook Solution of dataset to refer.

https://github.com/komalghatvilkar/Internship/blob/main/Malignant%20Comments%20Classifier%20Project/Malignant%20Comments%20Classifier%20Project.ipynb

Thank You................!!!