MALIGNANT COMMENTS CLASSIFIER PROJECT

Submitted by:

MIENGANDHA SINHA

ACKNOWLEDGEMENT

I would like to express my gratitude to the Company to give this project to me. In making of this project I hereby used to take help from the references which is given by the company as sample documentation and details related to project and professionals and SME guided me a lot in the project and the other previous projects helped me and guided me in completion of the project.

INTRODUCTION

- Malignant Comments Classifier Problem

The project aims towards the social media which enables people to express their opinions widely online. It passionately and relentlessly malevolent type of work which means aggressively malicious in nature which can hurt others sentiment. 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

A primary objective of this project is to build a model that can differentiate between comments and its categories by using some attributes that are highly correlated. As it is said in the statement that the malignant comments classifier case project, the data set is huge and includes many categories of comments, by doing data exploration and derive some interesting features using the comments text column available. Some research on this project, are able to train the model and predicting things make the efficient work.

- ## Review of Project

In this project, by doing some predictive research 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. From the collecting data phase gives important independent variables. The model building do all the data visualizations, data pre-processing steps, evaluating the model, data cleaning and selecting the best model for the project.

- ## Motivation for the Problem Undertaken

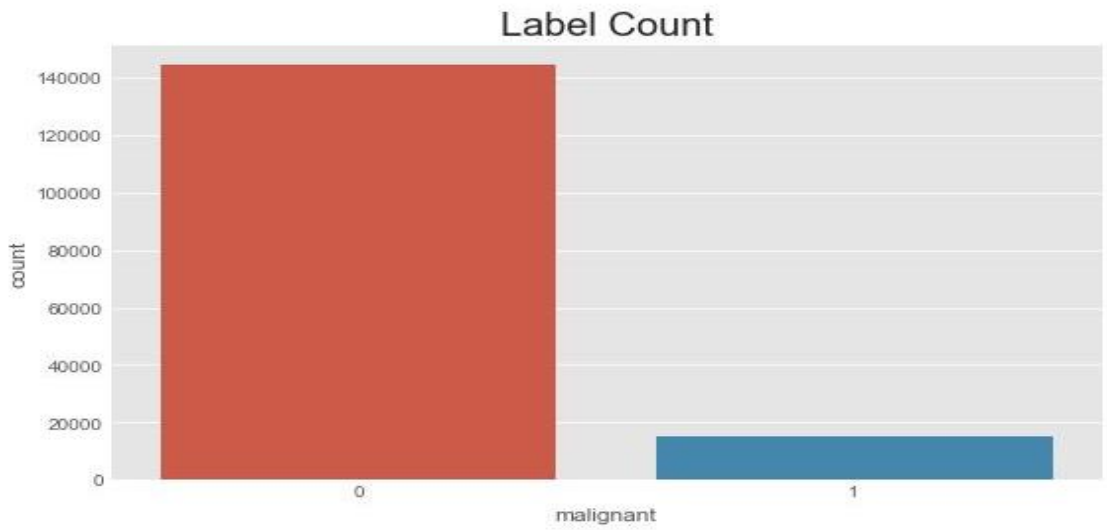
The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples, no null values, EDA has to be performed to see whether it gain or loss in the independent variable and its compare to the price among every aspects, to build machine learning models, to determine the optimal values of Hyper parameters and the selection of the best model. Here, 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.

ANALYTICAL PROBLEM FRAMING

- Mathematical/Analytical Modelling of the Problem

By checking the null values found that having no null values in the datasets, the type of data frame is in pandas, data frame info tells about the variables that train.csv has int64(6), object(2) and test.csv has object(2), by using the data visualization they are:

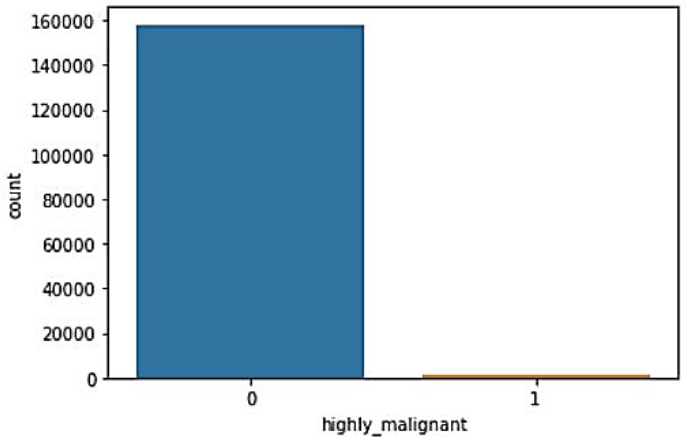
1. It is showing the label column, which includes values 0 and 1, denoting if the comment is malignant or not.



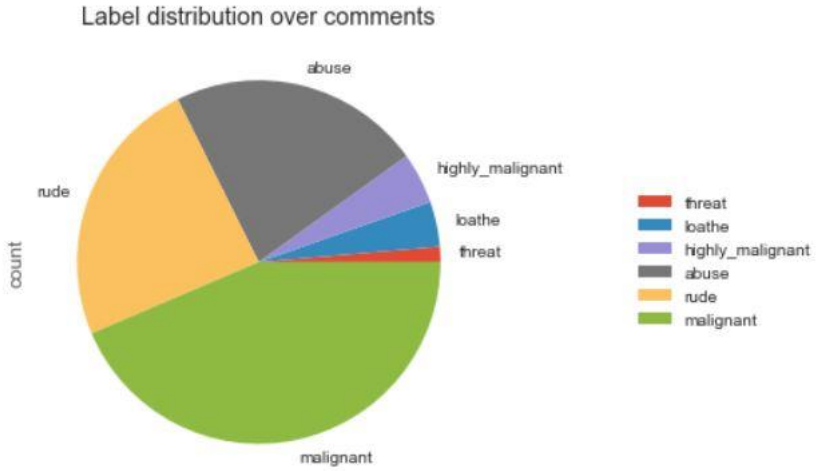
2. It denotes comments that are highly malignant and hurtful

highly_malignant

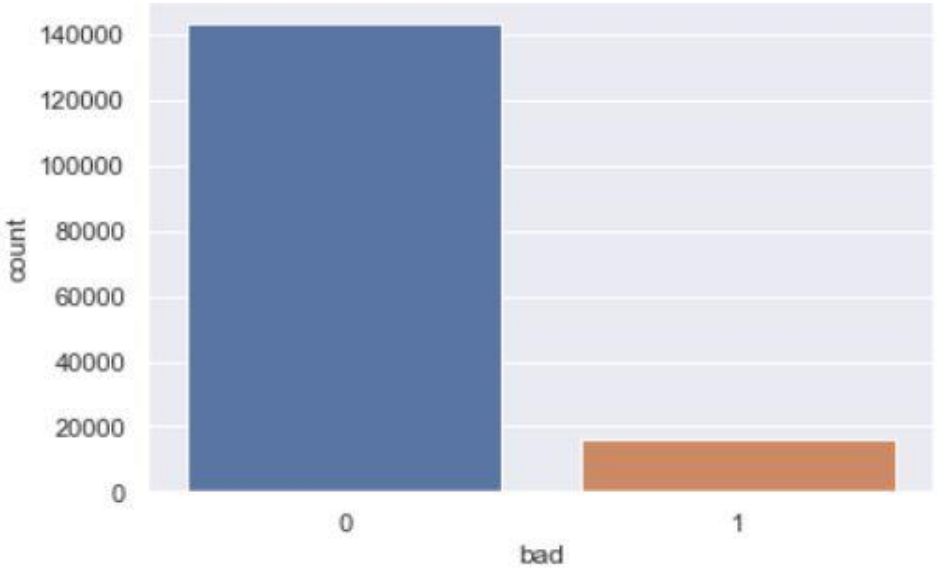
0 157976
1 1595
Name: highly_malignant, dtype: int64



3. Target the columns over distribution chart



4. It shows ‘how bad the comments are’



- Data Sources and their formats

The data sources and their formats are from .csv file.

Training dataset

```
df_train=pd.read_csv('train.csv')
df_train.head()
```

	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

Predicting dataset

```
df_test=pd.read_csv('test.csv')
df_test.head()
```

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

- Data Pre-Processing Done

The steps followed for the cleaning of the data after the importing pre-processing there transform the target columns into features then lastly it has to set for the data frame.

- ## Data Inputs-Logic-Output Relationships

  The relationships between inputs and outputs can be studied extracting weights of the trained model. Regression is that relationships between them can be blocky or highly structured based on the training data. It requires the data scientist to train the algorithm with both labeled inputs and desired outputs.

- ## State the set of assumptions(if any) related to the problem under consideration

  Presumptions are by using regression label encoding, classifier, selection of the best models, various machine learning to predict that it means the relationship between the dependent and independent variables look fairly linear. Thus, our linearity assumption is satisfied.

- ## Hardware and Software Requirements and Tools Used

  By importing many libraries are

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

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

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

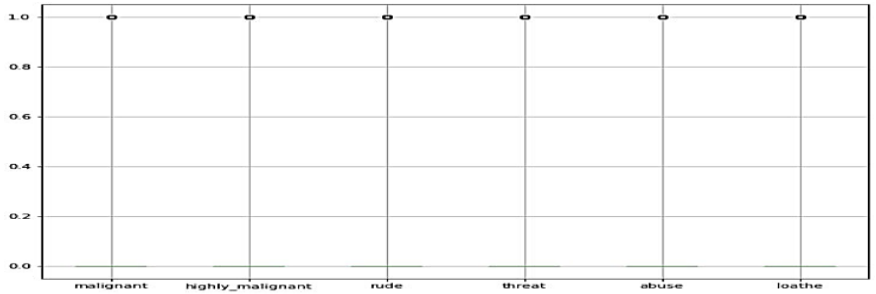
MODEL/s DEVELOPMENT AND EVALUATION

- Identification of possible problem-solving approaches(methods)

The collection and interpretation of data in order to uncover patterns and trends. It is a component of data analytics. Statistical analysis can be used in situations like gathering research interpretations, statistical modelling or designing surveys and studies. The approaches/methods of identification are descriptive and inferential statistics which are describes as the properties of sample and population data, and inferential statistics which uses those properties to test hypotheses and draw efficient conclusions in terms of outputs.

- Testing of Identified Approaches(Algorithms)

  There is no outliers



- Run and Evaluate selected models

  Data Cleaning

The dataset is relatively clean. There is a minor data leak, IP addresses appended to some comments. For various reasons, mainly that this data may slightly compromise the model's ability to generalize to new data, I've used a regular expression.

```
1 total=df_train.isnull().sum().sort_values(ascending=False)
2 percent = (df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=False)
3 missing = pd.concat([total, percent], axis=1, keys=['Total' , 'Percent'])
4 missing.head()
```

	Total	Percent
id	0	0.0
comment_text	0	0.0
malignant	0	0.0
highly_malignant	0	0.0
rude	0	0.0

Feature Engineering

The engineered features are normalized from 0.0 to 1.0. The tf-idf features are not scaled.

Model Evaluation

The model has been trained, tested, and optimized using training and test subsets of the data. I will use an unseen holdout subset of the data to evaluate the model.

Model Building

```
# our problem is classification type of problem.
# import useful libraries for machine learning algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curve,roc_auc_score,auc,f1_score
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score,GridSearchCV

model = [LogisticRegression(solver='liblinear'),DecisionTreeClassifier(),MultinomialNB()]

for m in model:
    m.fit(x_train,y_train)
    train = m.score(x_train,y_train)
    predm = m.predict(x_test)
    print("Accuracy of",m,"is:")
    print("Accuracy of training model is:",train)
    print("Accuracy Score:",accuracy_score(y_test,predm))
    print("Confusion matrix:","\n",confusion_matrix(y_test,predm))
    print("Classification report:","\n",classification_report(y_test,predm))
    print("\n")
```

```
Accuracy of LogisticRegression(solver='liblinear') is:
Accuracy of training model is: 0.9356101729648184
Accuracy Score: 0.9286524006348211
Confusion matrix:
 [[26180  2342]
 [ 1749 27068]]
Classification report:
              precision    recall  f1-score   support

           0       0.94      0.92      0.93     28522
           1       0.92      0.94      0.93     28817

    accuracy                           0.93     57339
   macro avg       0.93      0.93      0.93     57339
weighted avg       0.93      0.93      0.93     57339


Accuracy of DecisionTreeClassifier() is:
Accuracy of training model is: 0.9967866127759393
Accuracy Score: 0.9442962032822337
Confusion matrix:
 [[26356  2166]
 [ 1028 27789]]
Classification report:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94     28522
           1       0.93      0.96      0.95     28817

    accuracy                           0.94     57339
   macro avg       0.95      0.94      0.94     57339
weighted avg       0.94      0.94      0.94     57339


Accuracy of MultinomialNB() is:
Accuracy of training model is: 0.8938928202377994
Accuracy Score: 0.8929698808838661
Confusion matrix:
 [[25832  2690]
 [ 3447 25370]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.91      0.89     28522
           1       0.90      0.88      0.89     28817

    accuracy                           0.89     57339
   macro avg       0.89      0.89      0.89     57339
weighted avg       0.89      0.89      0.89     57339
```

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))
```

```
Training accuracy is 0.9449768398020265
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

Decision Tree Classifier

```
DTC = DecisionTreeClassifier()

DTC.fit(x_train, y_train)
y_pred_train = DTC.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = DTC.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))
```

```
Training accuracy is 0.9987652076895595
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

Random Forest Classifier

```
RFC=RandomForestClassifier()
RFC.fit(x_train, y_train)
y_pred_train = RFC.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RFC.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))
```

```
Training accuracy is 0.9987526717270169
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

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))
```

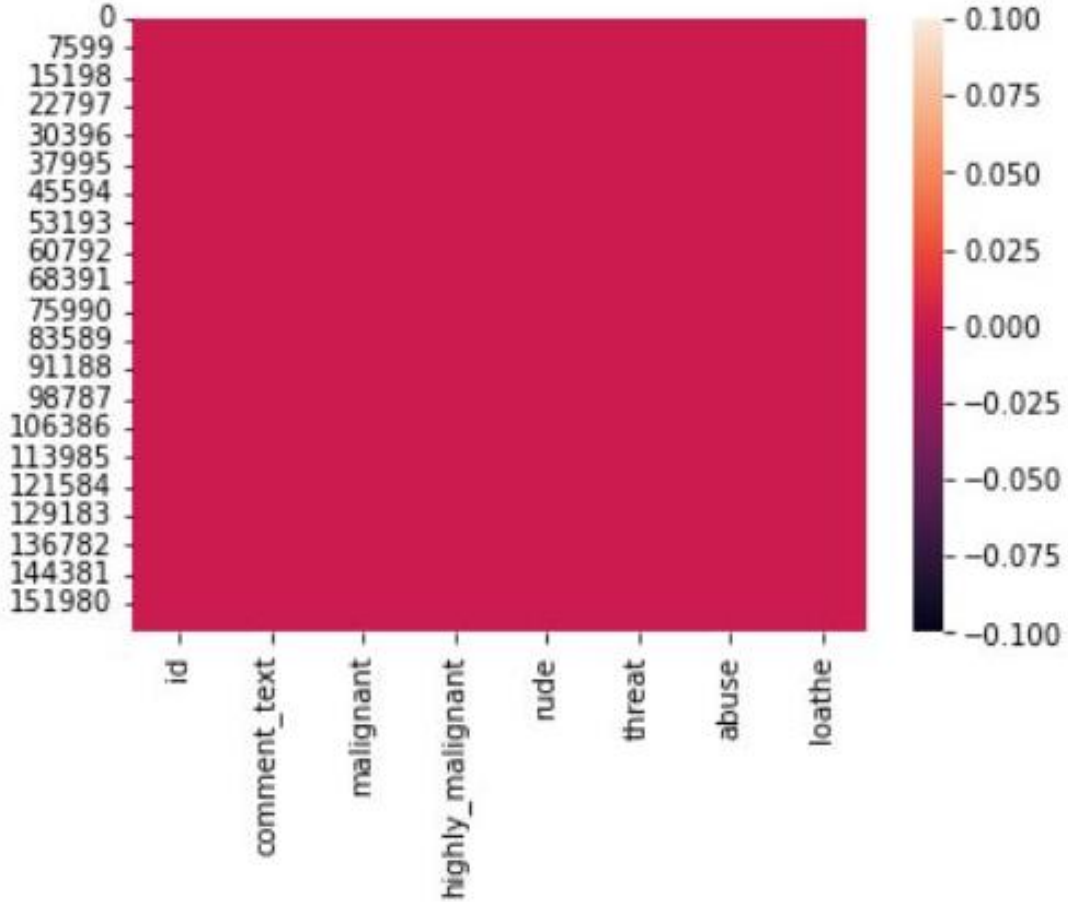
```
Training accuracy is 0.9603926263468324
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

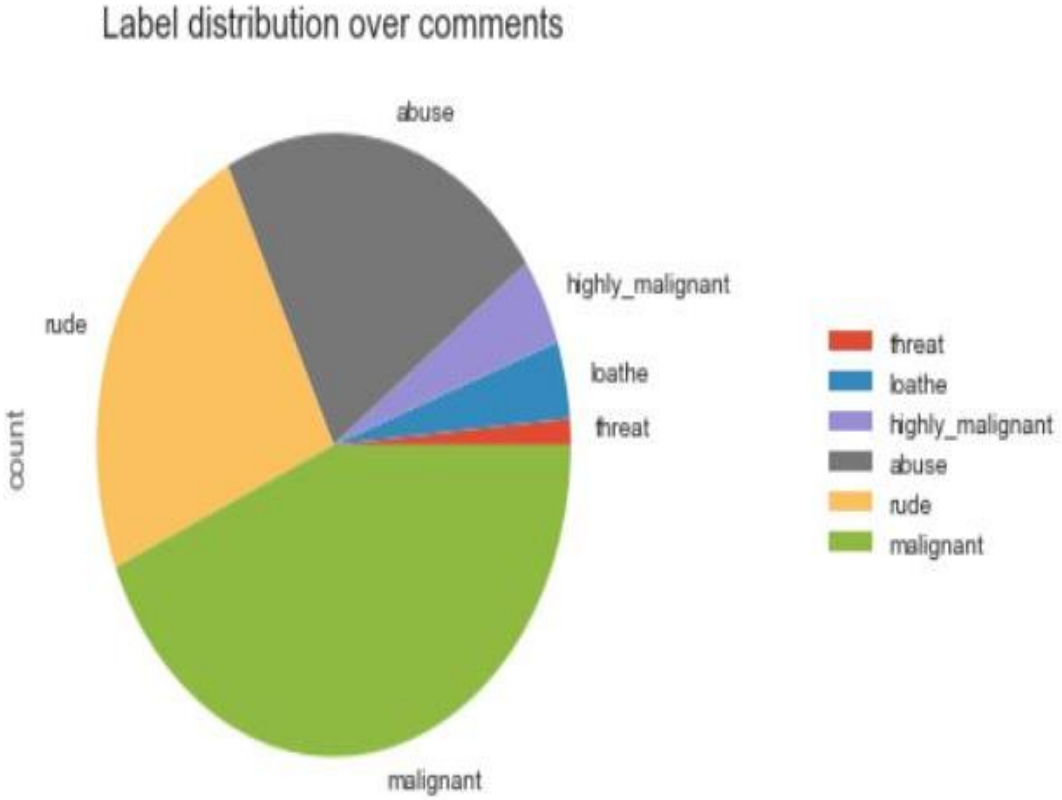
```
#RandomForestClassifier
RF =RandomForestClassifier()
RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
cvs=cross_val_score(RF, x, y, cv=10, scoring='accuracy').mean()
print('cross validation score :',cvs*100)
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9987464037457456
Test accuracy is 0.9333333333333333
cross validation score : 95.66399866799561
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

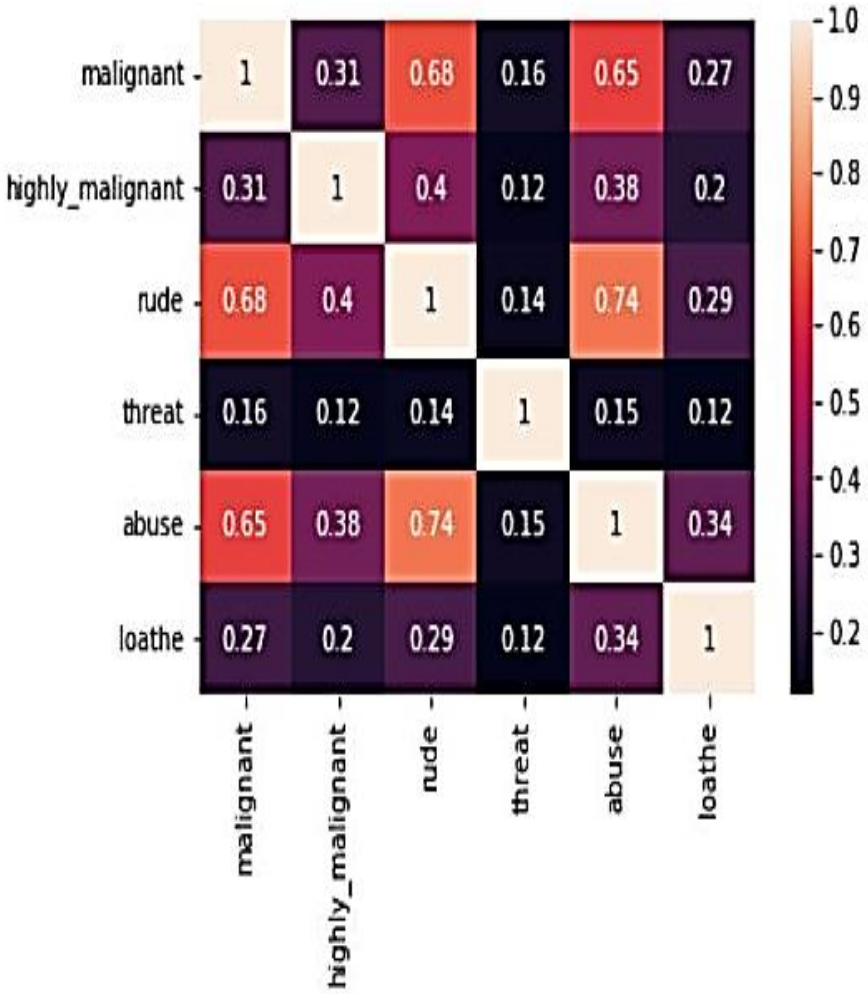
From the above, the best model randomforest classifier, which is 99%.

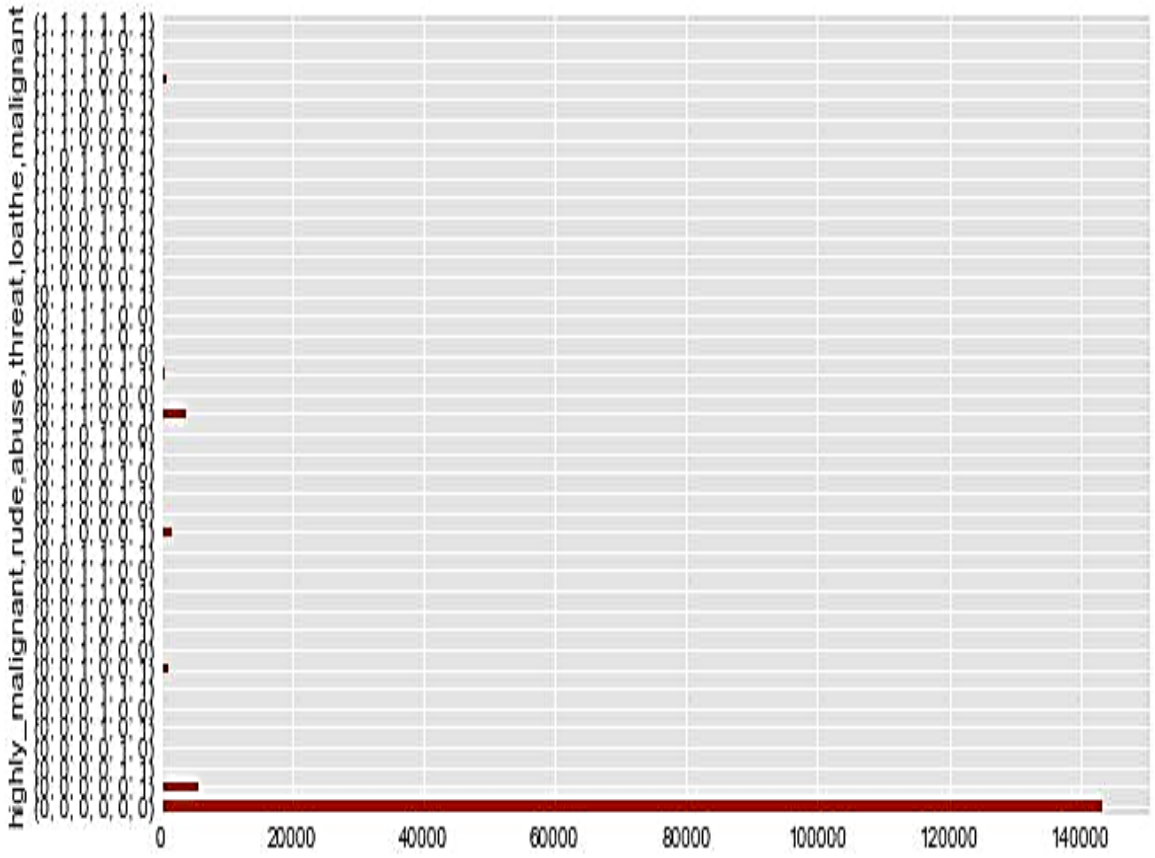
- Visualizations





Visualizing the relationships





The relationship between the dependent and independent variables look good in linear. Thus, our linearity assumption is satisfied.

• Interpretation of the Results

The final model of the independent variables and dependent variables are exactly vary with the variables and offers a significant performance boost over the logistic regression model, about accuracy of training model 93%. So far in the abstract about F1 scores. This model has 93% accuracy. The model performed at predicting a malignant comment is recall. It achieved a recall score of 0.60; 60% of the actual malignant comments as malignant. If we used recall as a training objective, it would classify every comment as malignant and reach 100% recall and make every clean comment a false positive. It's necessary then to strike a balance between precision and recall. False positives waste time, while fast negatives allow malignant to fall through the cracks. This model is robust enough for this application and it offers a large advantage over both the standard approach of human flagging for review and an out-of-the-box model. Of the comments would be submitted to a moderator review by the model, 60% are malignant. An effective tool that would both save moderators time and efficiently catch comments that may otherwise fall through the cracks, this is the most important and the most time consuming. Manually collecting data daily is efficient to do work with the research data. Each moderator could have a big impact on reducing malignant in the Wikipedia community.

CONCLUSION

I would like to conclude here that by doing research in the topic by through some points:

- Analyze the problem and purpose a useful solution
- Explore the dataset to get better picture of how the labels are distributed, how they correlate with each other
- Develop an objective that fits a practical use case and addresses the major class imbalance
- Create a baseline score with a simple logistic regression classifier
- Explore the effectiveness of multiple machine learning algorithms
- Select the best model based on a balance of performance and efficiency
- Tune the model parameters to maximize performance
- Build the final model with the best performing algorithm and parameters and test it on a holdout subset of the data
- And the final model offered about 93% performance gain over the initial benchmark model, which makes it an effective solution to the problem.
- By using multiple models, a sort of divide the conquer method where the problem is divided into multiple smaller, contextual problems. While the solution generalizes to the entire dataset, no one solution will be able to generalize perfectly to the diverse variety of inputs from Internet users. By training models on different situations, like a model that's only been trained on short or long comments, to only detect whether a comment is malignant when profanity is present by use a simple decision tree to feel comments into the model that would be most effective.

THANK YOU