MALIGNANT COMMENTS PROJECT

Submitted by:

PROMISE AZOM

ACKNOWLEDGMENT

I would like to express my appreciation to Datatrained Education and Fliprobo Technologies for their support in bringing me this far in my Data Science Journey. Your lectures and videos made the difference.

I also want to thank my wife and family for their patience.

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. 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. 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.

  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.

• Conceptual Background of the Domain Problem

This is critical stage in any machine learning process. It involves brainstorming and coming up with as many hypotheses as possible about what could affect the target variable. It facilitates in exploring the data at hand more efficiently and effectively. Domain Knowledge should be done before seeing the data or else we will end up with biased hypotheses. Below are some anticipated assertions on the problem statement.

Social media has been a tool for a people who have an excessive interest in or admiration of themselves but the fact of the matter is, they have always existed. Social media has just given them an extra tool to terrorize people. Here are five ways – those without empathy and with an excessive sense of entitlement – use social media to exploit, manipulate and destroy their victims:

1. To triangulate.

Social media is a veritable playground for malignant users. It gives them easy access to multiple victims and the ability to manufacture love triangles in covert and insidious ways.

2. To infiltrate.

They want to know anything and everything about you, so that they can later use your wounds against you. Having access to your social media accounts can give them an easy way to find out more about your likes, interests, hobbies, and desires. Remember, it's possible for even a complete stranger to find out your life story should they do the due diligence of looking through your photos etc

3. To stalk and harass you.

Even if you block them, they can make fake social media accounts to 'check-up' on your whereabouts.

4. To self-aggrandize.

Recent study shows that those with an inherent belief in their own superiority are more likely to be found glorifying themselves on social media as opposed to more vulnerable narcissists with lower self-esteem.

5. To bully and taunt.

Research reveals that online trolls possess the Dark Tetrad traits of narcissism, psychopathy and Machiavellianism. In other words, online narcissists take sadistic pleasure in provoking others.

- **Review of Literature**
  The below key operations will be adopted:
  - Exploratory Data Analysis (EDA)
  - Data Preprocessing (Univariate,Bivariate,Multivariate)
  - Feature Selection
  - Metrics Measurement
  - Model Execution
  - Hyperparameter Tunning
  - Model Saving

- **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 Modeling of the Problem Mathematical Modelling:
  - Update of null values
  - Encoding

  Statistical

  Modelling:

  - Variance Inflation factor
  - ANOVA Analytical

  Modelling:

- Visualization Techniques: Density Plot, Scatter Plot,Count Plot, Boxplot

- **Data Sources and their formats**
  - Two datasets are being provided (test.csv, train.csv).
  - Train Data contains 159571 entries each having 8 variables (Target variable inclusive).
  - Test Data contains 153,164 entries each having 80 variables (Target variable exclusive).
  - Data does not contains Null values.
  - Data contains numerical as well as categorical variable.

See below table showing summary of the Variables:

S/N	Variable	Variable Type	Data Type	Data Form	Null	Variable Attributes
1	Id	Categorical	Object	Non-Numerical	0	It includes unique Ids associated with each comment text given
2	comment_text	Categorical	Object	Non-Numerical	0	This column contains the comments extracted from various social media platforms

3	Highly Malignant	Categorical	int64	Numerical	0	It denotes comments that are highly malignant and hurtful
4	Rude	Categorical	int64	Numerical	0	It denotes comments that are very rude and offensive
5	Threat	Categorical	int64	Numerical	0	It contains indication of the comments that are giving any threat to someone
6	Abuse	Categorical	int64	Numerical	0	It is for comments that are abusive in nature
7	Loathe	Categorical	int64	Numerical	0	It describes the comments which are hateful and loathing in nature
8	malignant	Categorical	int64	Numerical	0	It is the Label column, which includes values 0(NO) and 1(YES), denoting if the comment is malignant or not (target variable)

NB: The Malignant is the Dependent Variable (Target/Label) while the rest are the independent variables (Features).

Also see snap shots of data…

Full view of Data

	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

First Five rows

	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

Last Five rows

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
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

Random Four samples

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
146898	3367e06f6f2b3a6d	Convert to Puritanism or die? \n\nThe bit abou...	0	0	0	0	0	0
145283	1a2843b373407cc5	hello again \n\nhello, do you remember the van...	0	0	0	0	0	0
69486	b9d936c13eb9bdee	If one can read english, we can see the verse ...	0	0	0	0	0	0
21665	391a4c24561183a6	"::""There were also several aerial bombardmen...	0	0	0	0	0	0

Data Description(Continous Data)

	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

Data Description(Categorical Data)

	id	comment_text
count	159571	159571
unique	159571	159571
top	0000997932d777bf	Explanation\nWhy the edits made under my usern...
freq	1	1

- **Data Pre-processing Done**
  Observations and Assumptions on Data Cleaning
  Observations
  No null values

Assumptions

All features are have no null values..hence needless of data cleansing!

1. Encoding

- We are fully aware we cannot run an exhaustive EDA on non-numerical data.

- This makes it necessary for us to convert all non-numerical data into numerical ones.

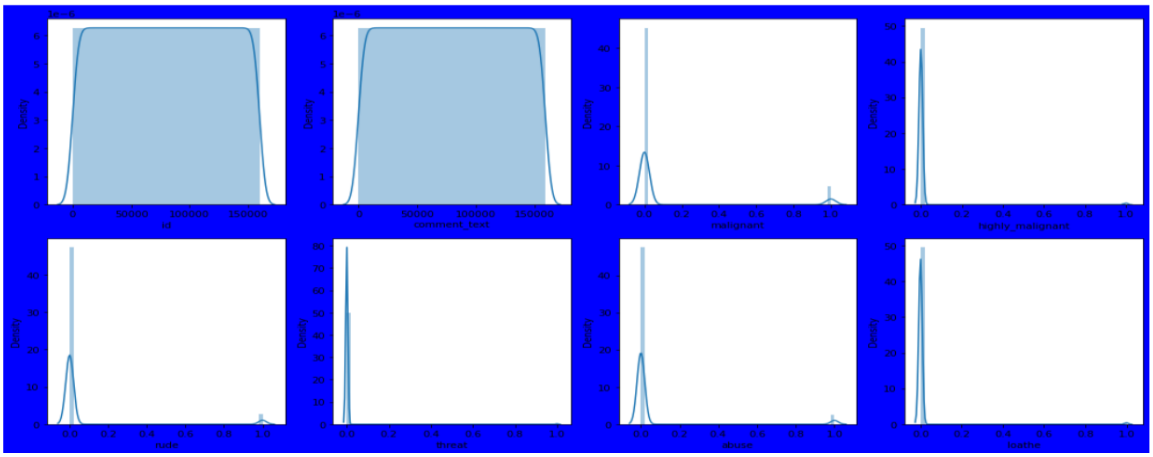
- Hence Label encoding was used on all the object columns.

Encoded data

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0	72698	0	0	0	0	0	0
1	1	68359	0	0	0	0	0	0
2	2	79594	0	0	0	0	0	0
3	4	35519	0	0	0	0	0	0
4	5	146426	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	159505	44289	0	0	0	0	0	0
159567	159510	145720	0	0	0	0	0	0
159568	159524	122203	0	0	0	0	0	0
159569	159535	60037	0	0	0	0	0	0
159570	159541	31106	0	0	0	0	0	0

159571 rows × 8 columns

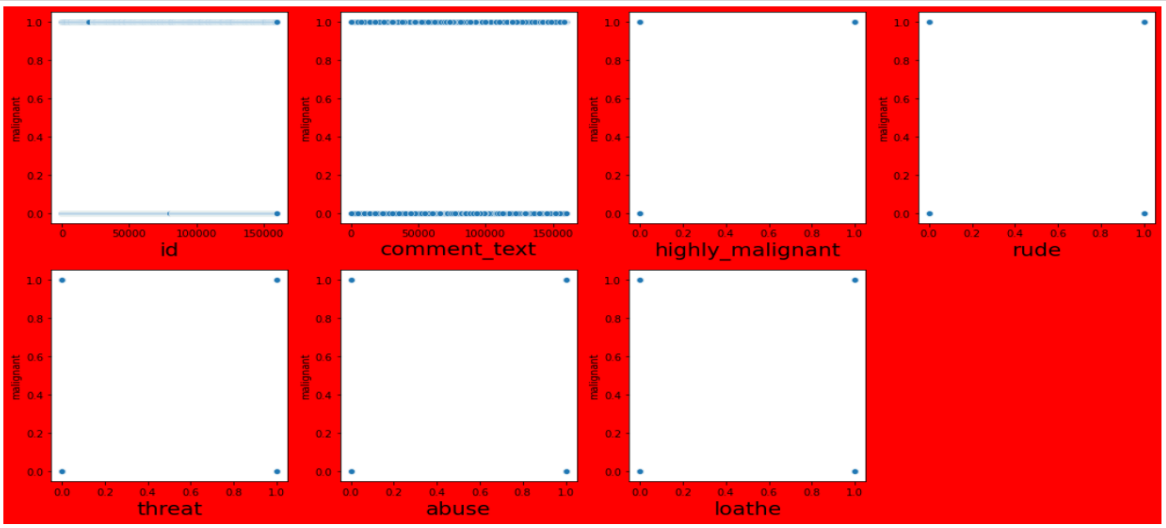
- **Data Inputs- Logic- Output Relationships**
  Normal Distribution Check(Univariate Analysis)
  From a density plot standpoint:



- All the features does not obey a normal distribution, the From the above density plot we can see that majority of features does not obey a normal distribution, the building blocks are not in tandem with a normalized curve:

Scatter Plot Check(Bivariate Analysis)

From a Scatter plot standpoint:



From the above scatter plot we can see a strong relationship between some features and the target(label)! Wow thats great! ...making our feature selection process less difficult as long as multicolinearity does not exist!

Correlation Check (Collinearity and Multicollinearity)- Multivariate Analysis;

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
id	1.000000	0.002812	-0.003263	-0.001403	-0.002188	-0.001165	-0.002086	-0.000844
comment_text	0.002812	1.000000	0.132016	0.057627	0.104020	0.026093	0.111724	0.046234
malignant	-0.003263	0.132016	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009
highly_malignant	-0.001403	0.057627	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600
rude	-0.002188	0.104020	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867
threat	-0.001165	0.026093	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128
abuse	-0.002086	0.111724	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736
loathe	-0.000844	0.046234	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000

From the above correlation statistics;

Collinearity:

- id has a negative correlation of 0.0% with the target column which can be considered as NO BOND

- comment_text has a positive correlation of 0.13% with the target column which can

be considered as a fair bond

- threat has a positive correlation of 0.16% with the target column which can be considered as fair BOND

- loathe has a positive correlation of 0.27% with the target column which can be considered as a good BOND

- highly_malignant has a positive correlation of 0.31% with the target column which can be considered as a good BOND

- abuse has a positive correlation of 0.65% with the target column which can be considered as very good BOND

- rude has a positive correlation of 0.68% with the target column which can be considered as very good BOND

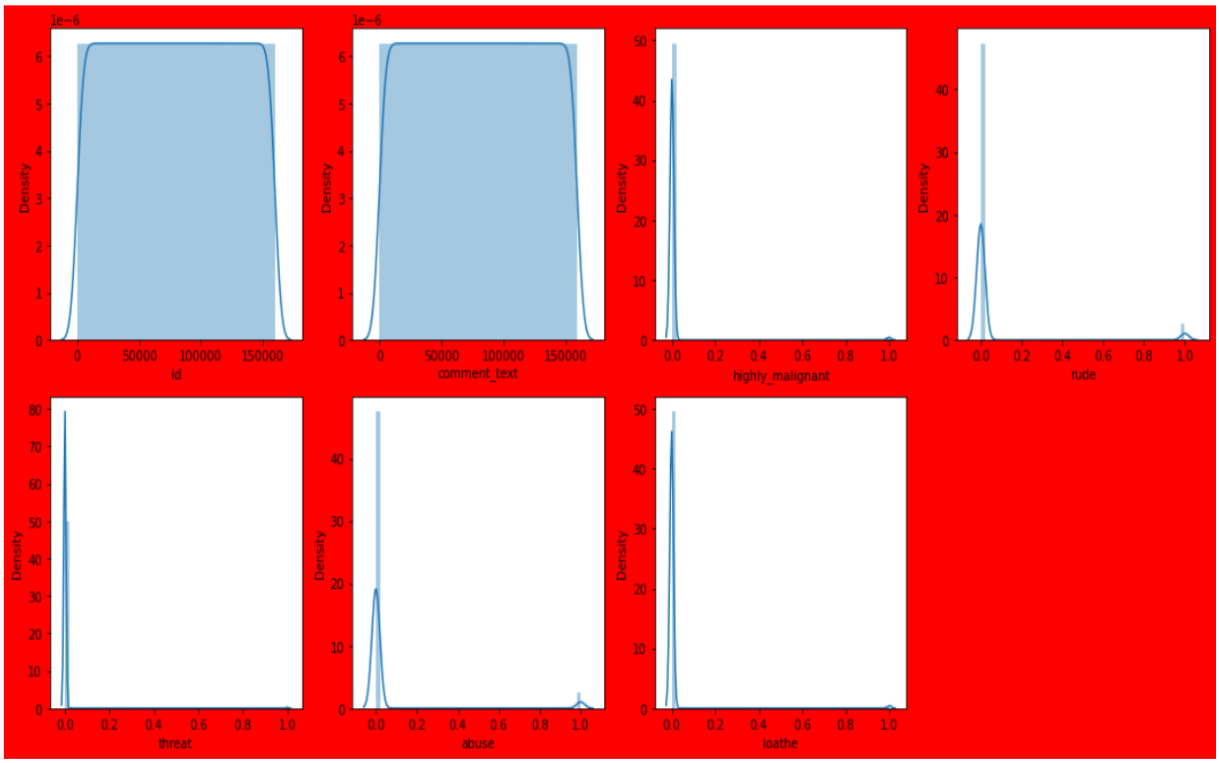
- Feature with Maximum correlation = rude
- Features with Minimum correlation = id

Multicollinearity:
- From the heatmap we can see that some of features have some noticeable correllation between them
But we will reconfirm during further steps by finding the Variance Inflation Factor(VIF)

NB: Multicollinnearity means two variables are explaining the same thing, meaning one of them is useless and therefore we have to drop one of them!

Skewness Check

We assumed a Skewness threshold is taken as +/-0.50. Meaning any value outside +/-0.50 contains skewness. Hence some of the features are having a skewness:




```
id:
7.386055e-17

comment_text:
1.282301e-19

highly_malign
ant:
9.851722e+00

rude:
3.992817e+00

threat:
1.818900e+01

abuse:
4.160540e+00

loathe:
1.051592e+01
```

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

  Assumption on Variance Inflation Factor

  We used a variance inflation Factor of 5. Meaning any feature with Variance Inflation Factor greater than 10 is assumed to have a multicollinearity problem. It is not standard. The dataset demands.

  In lieu of the above assumption, we will further drop the following:

  - GrLivArea
  - 1stFlrSF
  - TotalBsmtSF
  - 2ndFlrSF
  - Assumption on Skewness

    Lets assume Skewness threshold is taken as +/-0.5. Meaning any value outside +/-0.5 contains skewness. Hence all the Columns are having skewness:

- **Hardware and Software Requirements and Tools Used**

  Hardware Requirments

  - Device name: DESKTOP
  - ProcessorIntel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 2.90GHz
  - Installed RAM: 8.00 GB

- System type: 64-bit operating system, x64-based processor

Software Requirments

- Jupyter Notebook
- Python Programming Language
- Libraries
    i. Pandas: Used loading the dataset
    ii. Numpy: Used for rounding up numbers
    iii. sklearn.linear_model: Used for initializing the Logistic Regression Model
    iv. sklearn.neighbors: Used for initializing the KNeighbours Classifier Model
    v. sklearn.tree: Used for initializing the DecisionTree Classifier Model
    vi. sklearn.ensemble: Used for initializing the ensemble Techniques/Models - RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier,ExtraTreesClassifier
    vii. sklearn.preprocessing: Used Scaling, Power Transformation and Encoding of data
    viii. sklearn.model_selection: Used Data split into test and train. Also used for Initializing GridsearchCV during hyperparameter tunning
    ix. sklearn.metrics: Used for metrics measurment
    x. xgboost: Used for initializing the XGBoost Model
    xi. statsmodels: Used to solve for multicollinearity via Variance inflation Factor
    xii. Scipy.stats: Used to remove outliers via zscore
    xiii. Seaborn: Used for visualization during variate analysis

xiv. Matplotlib: Also used for visualization during variate analysis

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)
  The algorithm used were:
  - Standard Scaler
  - train_test_split
  - fit(x_train,y_train)

- Run and Evaluate selected models
  Eight Models were used:

  - Logistic Regression

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
pred_test=lr.predict(x_test)
pred_train=lr.predict(x_train)
Test_Accuracy_lr= (accuracy_score(y_test,pred_test))
Train_Accuracy_lr= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  96.07
Confusion Matrix  [[28741   161]
 [ 1093  1920]]
Classification Report               precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.92      0.64      0.75      3013

    accuracy                           0.96     31915
   macro avg       0.94      0.82      0.87     31915
weighted avg       0.96      0.96      0.96     31915
```

LogisticRegression is producing good accuracy 96%. Now we will check Cross Validation score as well for overfitting(if exists).

- KNeighbors Classifier

```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
pred_test=knn.predict(x_test)
pred_train=knn.predict(x_train)
Test_Accuracy_knn= (accuracy_score(y_test,pred_test))
Train_Accuracy_knn= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  95.92
Confusion Matrix  [[28693   209]
 [ 1092  1921]]
Classification Report                precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.90      0.64      0.75      3013

    accuracy                           0.96     31915
   macro avg       0.93      0.82      0.86     31915
weighted avg       0.96      0.96      0.96     31915
```

KNeighbors is producing good accuracy 96%. Now we will check Cross Validation score as well for overfitting(if exists).

- GradientBoosting Classifier

```
gb=GradientBoostingClassifier()
gb.fit(x_train,y_train)
pred_test=gb.predict(x_test)
pred_train=gb.predict(x_train)
Test_Accuracy_gb= (accuracy_score(y_test,pred_test))
Train_Accuracy_gb= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  96.11
Confusion Matrix  [[28733   169]
 [ 1074  1939]]
Classification Report               precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.92      0.64      0.76      3013

    accuracy                           0.96     31915
   macro avg       0.94      0.82      0.87     31915
weighted avg       0.96      0.96      0.96     31915
```

GradientBoosting Classifier is producing good accuracy = 96%. Now we will check Cross Validation score as well for overfitting(if exists).

- XGBoost Classifier

```
xgb=XGBClassifier()
xgb.fit(x_train,y_train)
pred_test=xgb.predict(x_test)
pred_train=xgb.predict(x_train)
Test_Accuracy_xgb= (accuracy_score(y_test,pred_test))
Train_Accuracy_xgb= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  96.08
Confusion Matrix  [[28733   169]
 [ 1081  1932]]
Classification Report               precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.92      0.64      0.76      3013

    accuracy                           0.96     31915
   macro avg       0.94      0.82      0.87     31915
weighted avg       0.96      0.96      0.96     31915
```

XGBoost Classifier is producing good accuracy = 96%. Now we will check Cross Validation score as well for overfitting(if exists).

- ExtraTrees Classifier

```
ex=XGBClassifier()
ex.fit(x_train,y_train)
pred_test=ex.predict(x_test)
pred_train=ex.predict(x_train)
Test_Accuracy_ex= (accuracy_score(y_test,pred_test))
Train_Accuracy_ex= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  96.08
Confusion Matrix  [[28733   169]
 [ 1081  1932]]
Classification Report                precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.92      0.64      0.76      3013

    accuracy                           0.96     31915
   macro avg       0.94      0.82      0.87     31915
weighted avg       0.96      0.96      0.96     31915
```

Extratrees Classifier is producing good accuracy = 96%. Now we will check Cross Validation score as well for overfitting(if exists).

- Support Vector Classifier

```
sv=SVC()
sv.fit(x_train,y_train)
pred_test=sv.predict(x_test)
pred_train=sv.predict(x_train)
Test_Accuracy_sv= (accuracy_score(y_test,pred_test))
Train_Accuracy_sv= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  96.11
Confusion Matrix  [[28731   171]
 [ 1069  1944]]
Classification Report                precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.92      0.65      0.76      3013

    accuracy                           0.96     31915
   macro avg       0.94      0.82      0.87     31915
weighted avg       0.96      0.96      0.96     31915
```

Support Vector is producing good accuracy = 96%. Now we will check Cross Validation score as well for overfitting(if exists).

- MLP Classifier

```
mlp=MLPClassifier()
mlp.fit(x_train,y_train)
pred_test=mlp.predict(x_test)
pred_train=mlp.predict(x_train)
Test_Accuracy_mlp= (accuracy_score(y_test,pred_test))
Train_Accuracy_mlp= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  96.11
Confusion Matrix  [[28731   171]
 [ 1071  1942]]
Classification Report                precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.92      0.64      0.76      3013

    accuracy                           0.96     31915
   macro avg       0.94      0.82      0.87     31915
weighted avg       0.96      0.96      0.96     31915
```

MLP Classifier is producing good accuracy = 96%. Now we will check Cross Validation score as well for overfitting(if exists).

- GuassianNB Classifier

```
nb=GaussianNB()
nb.fit(x_train,y_train)
pred_test=nb.predict(x_test)
pred_train=nb.predict(x_train)
Test_Accuracy_nb= (accuracy_score(y_test,pred_test))
Train_Accuracy_nb= (accuracy_score(y_train,pred_train))
print("Test_Accuracy ",round(accuracy_score(y_test,pred_test)*100,2))#testing score
print("Confusion Matrix ", confusion_matrix(y_test,pred_test))
print("Classification Report ", classification_report(y_test,pred_test))
```

```
Test_Accuracy  96.11
Confusion Matrix  [[28731   171]
 [ 1069  1944]]
Classification Report                precision    recall  f1-score   support

           0       0.96      0.99      0.98     28902
           1       0.92      0.65      0.76      3013

    accuracy                           0.96     31915
   macro avg       0.94      0.82      0.87     31915
weighted avg       0.96      0.96      0.96     31915
```

GaussianNB is producing good accuracy 96%. Now we will check Cross Validation score as well for overfitting(if exists).

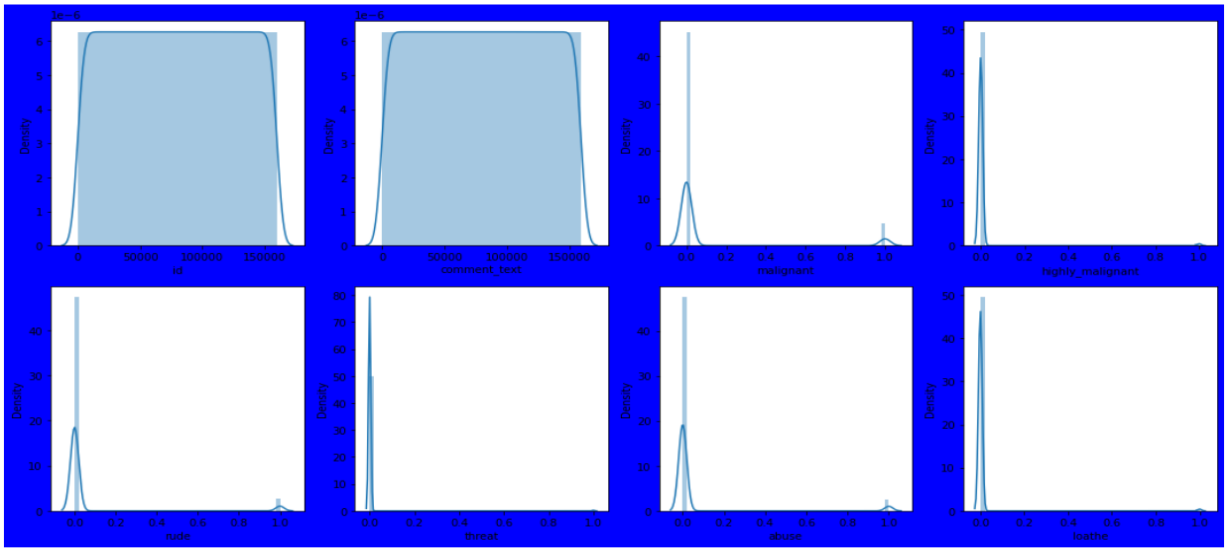
- **Key Metrics for success in solving problem under consideration**
- Accuracy score
- Confusion Matrix
- Recall
- Precision
- F1Score

- Cross validation
- AUC Score

The above metrics were used since it’s a Classification Problem

- **Visualizations**

<u>Normal Distribution Check(Univariate Analysis)</u>

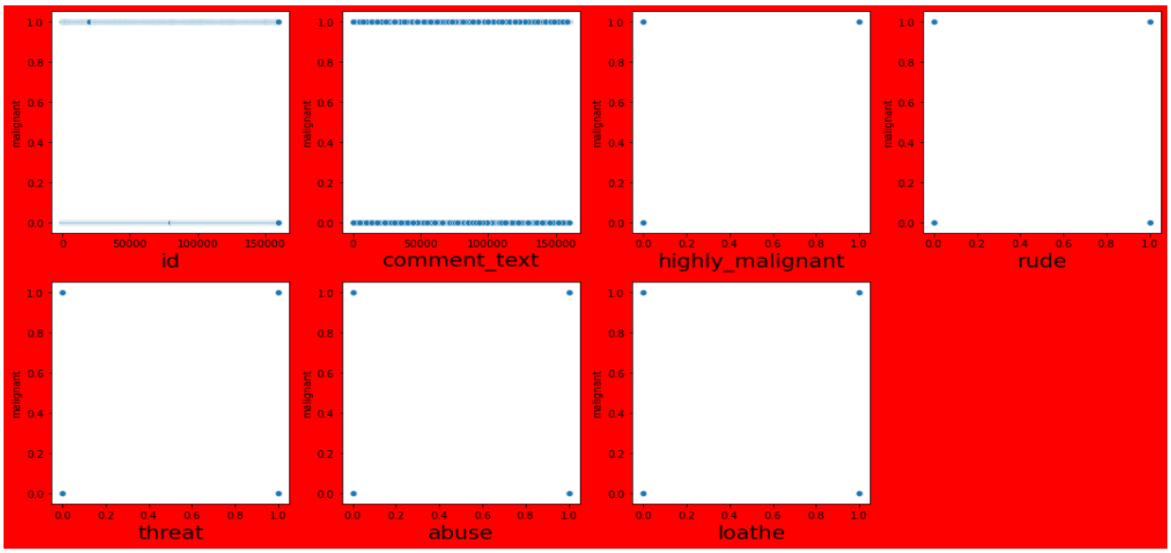


<u>Observations on Normal Distribution Check</u>

From the above density plot we can see that majority of features does not obey a normal distribution, the building blocks are not in tandem with a normalized curve:

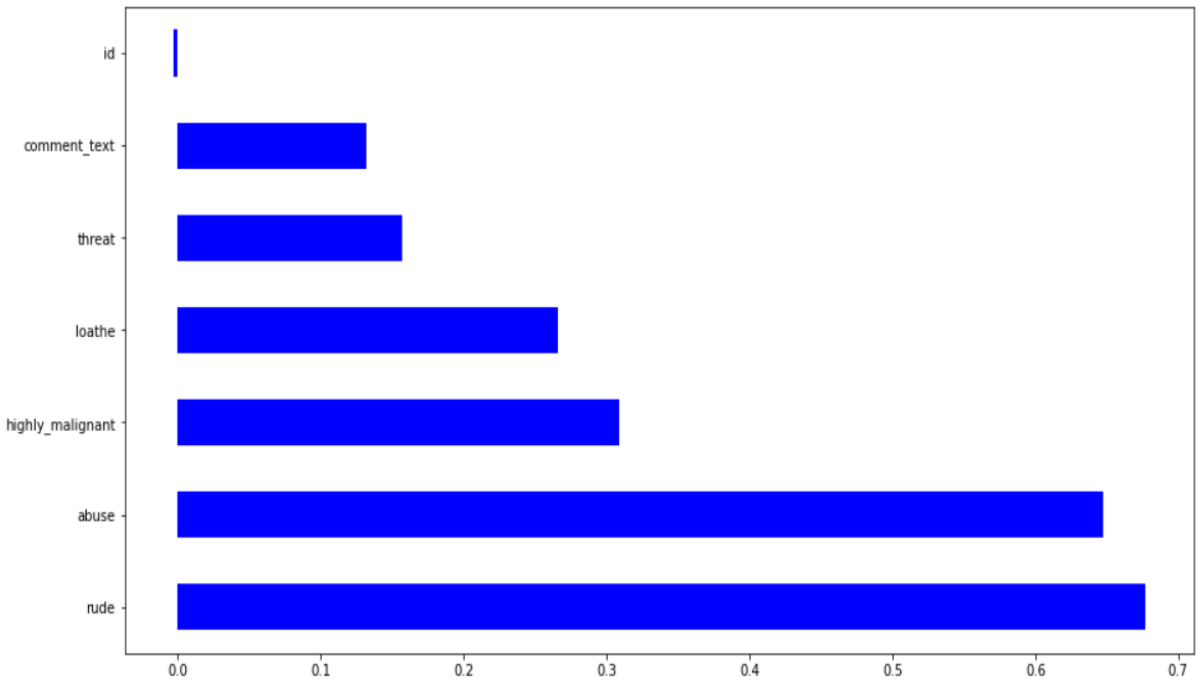
*The normal distribution of the 'malignant' has no contribution to our Model Building

Scatter Plot Check(Bivariate Analysis)



From the above scatter plot we can see a strong relationship between some features and the target(label)! Wow thats great! ...making our feature selection process less difficult as long as multicolinearity does not exist!

Correlation Check(Collinearity and Multicollinearity)- Multivariate Analysis



Observations

- id has a negative correlation of 0.0% with the target column which can be considered as NO BOND

- comment_text has a positive correlation of 0.13% with the target column which can be considered as a fair bond

- threat has a positive correlation of 0.16% with the target column which can be considered as fair BOND

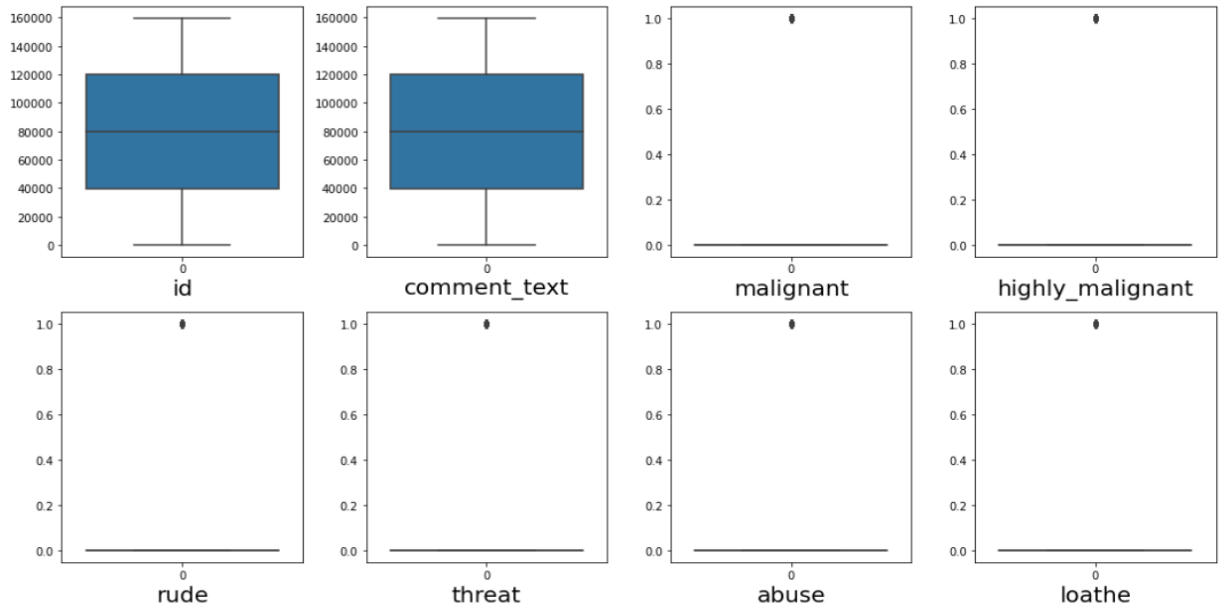
- loathe has a positive correlation of 0.27% with the target column which can be considered as a good BOND

- highly_malignant has a positive correlation of 0.31% with the target column which can be considered as a good BOND

- abuse has a positive correlation of 0.65% with the target column which can be considered as very good BOND

- rude has a positive correlation of 0.68% with the target column which can be considered as very good BOND

Outlier Check

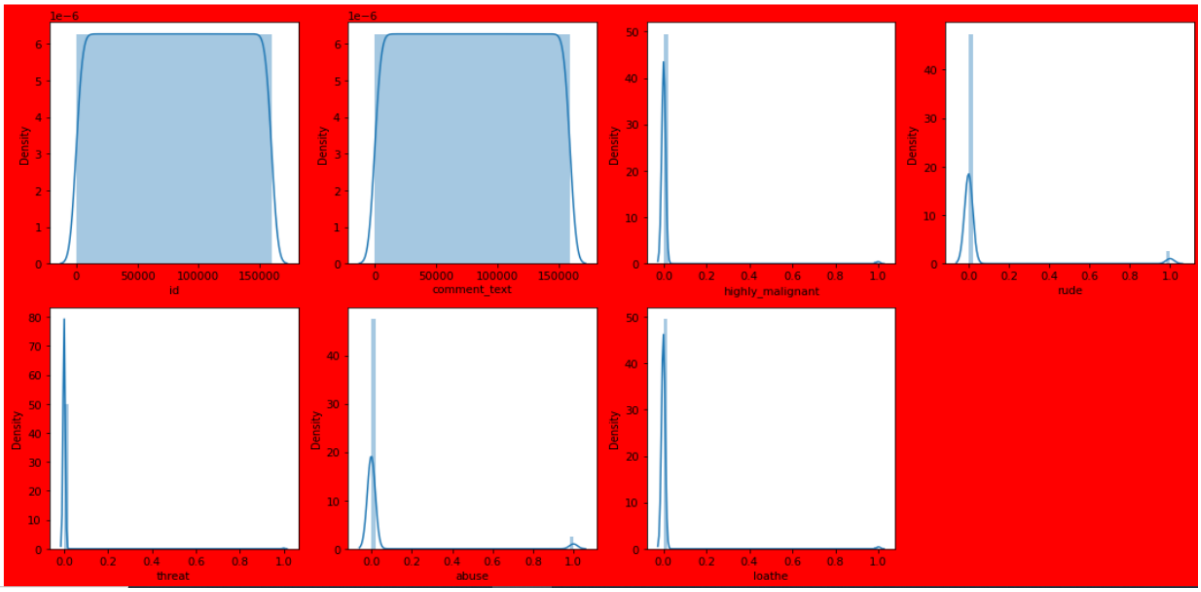


Observations on Outlier Check

From the above visualization plot its evident that there are outliers do exist!;

However this is subject to further analysis and reconfirmation using the zscore!,

Skewness Check



Observations on Skewness Check:

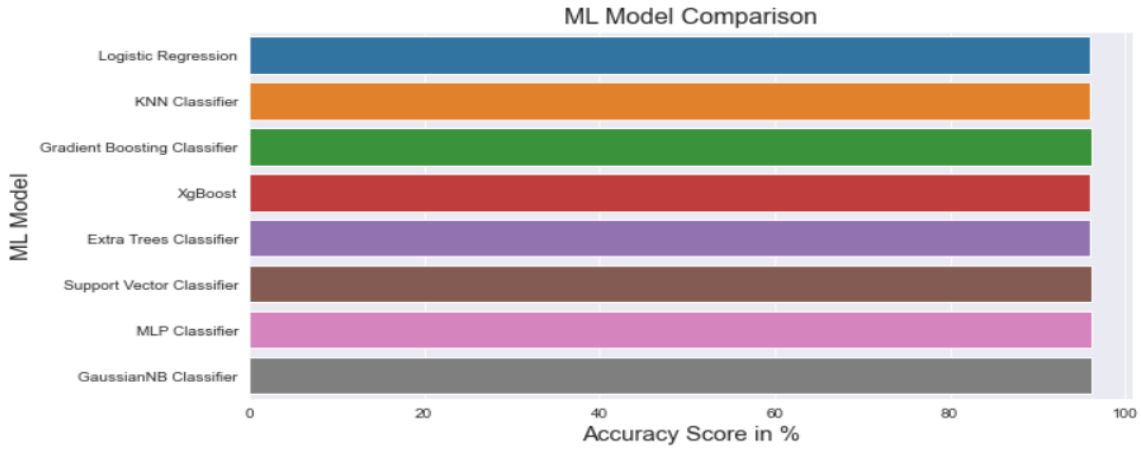
We assumed a Skewness threshold of +/-0.50. Meaning any value outside +/-0.50 contains skewness. Hence all of the features are having a skewness at first. However, after removal of outliers using zscore, we observed skewness has been eliminated by virtue of skewness removal!

CONCLUSION

- **Key Findings and Conclusions of the Study**

  COMPARING ALL EIGHT MACHINE LEARNING MODELS

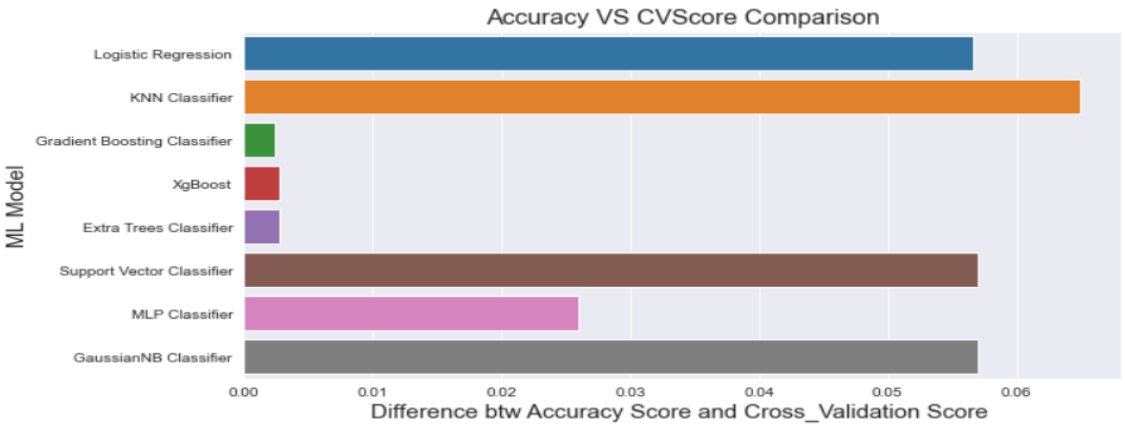
	Model	Accuracy_Score	Cross_Validation_Score	Accuracy_VS_CVScore
2	Gradient Boosting Classifier	96.11	95.86	0.002458
5	Support Vector Classifier	96.11	90.42	0.056991
6	MLP Classifier	96.11	93.52	0.025914
7	GaussianNB Classifier	96.11	90.42	0.056991
3	XgBoost	96.08	95.81	0.002727
4	Extra Trees Classifier	96.08	95.81	0.002727
0	Logistic Regression	96.07	90.42	0.056553
1	KNN Classifier	95.92	89.44	0.064850



Now from the above diagram it seems we have four models; Gradient Boosting, Support Vector, MLP and Guassian Classifiers producing the Highest Accuracy (96%), However, our aim is to find the BEST MODEL, so if we consider the difference Between Accuracy_Score and Cross_Validation_Score....

Comparing Differences between Accuracy and Cross_Validation Scores...

	Model	Accuracy_Score	Cross_Validation_Score	Accuracy_VS_CVScore
2	Gradient Boosting Classifier	96.11	95.86	0.002458
3	XgBoost	96.08	95.81	0.002727
4	Extra Trees Classifier	96.08	95.81	0.002727
6	MLP Classifier	96.11	93.52	0.025914
0	Logistic Regression	96.07	90.42	0.056553
5	Support Vector Classifier	96.11	90.42	0.056991
7	GaussianNB Classifier	96.11	90.42	0.056991
1	KNN Classifier	95.92	89.44	0.064850



From the above we can see the Model with least difference is Gradient Boosting Classifier!

Conclusion on Best Choice of Model

From the above we can see:

- The least difference is 0.002(very neglible)!

- The Model with least difference is Gradient Boosting Classifier!
- Accuracy is 96%

- We also went further to engage in **Hyperparamter Tunning** using GridSearchCV and arrived at a Final Accuracy of 96.11%

- Learning Outcomes of the Study in respect of Data Science
  - The removal of outliers played a major role in the Data Pre-processing.
  - The use of Variance Inflation Factor assured us that Multicollinearity does not exist and hence needless in dropping any features
- **Limitations of this work and Scope for Future Work**

  I was unable to predict the test data because my model was built with 7 features while the test data has only has 2 features. The Gradient Boosting Classifier is expecting 7 features(not 2 features) from the test data.(See Step 10 – Prediction with Test Data in Jupyter note book)