FLIP ROBO

Ratings-Prediction-Project

Submitted by:

HARPAL SINGH

ACKNOWLEDGMENT

The project is based on Ratings-Prediction machine building model. Various techniques used in the pipeline of Machine Learning model building is used from various sources like geeksforgeeks, seaborn documentation, towardsdatascience, medium, anaticsvidhya, kaggle website has been used. pandas library, NumPy library of python has been used. Scikit-learn machine learning models have been used for classification of target feature. Jupyter notebook has been used throughout the project and its various libraries have been called for various operations.

INTRODUCTION

Business Problem Framing

In this problem a client of the company is having a website who wants to add new feature to his website that the reviewer will have to add stars(rating) as well as the review. The rating will be out of 5 stars and it has only 5 options available 1 star, 2 stars, 3 stars, 4 stars and 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don’t have a rating. So, we have to build an application which can predict the rating by seeing the review.

Conceptual Background of the Domain Problem

Concept wise the problem of review and rating is very crucial for an e-commerce company to increase its sales thus increasing revenue. So, to solve this problem of prediction of rating according to reviews can be solved by machine learning using NLP and help the companies to increase their business.

Review of Literature

Various articles and reports from website like Kaggle, analyticsvidhya, medium and towardsdatascience have been used for reference and various techniques in text-pre-processing, data visualization and machine learning model building have been used from these websites.

Motivation for the Problem Undertaken

Objective to build this machine learning model is to have a hands on the model building techniques along with new facing new challenges while solving various anomalies in the dataset. And solving issues encountered during the machine learning building model. Since the number of records is quite high so computation takes a lot of time along with careful selection of features during feature engineering also posed some challenges.

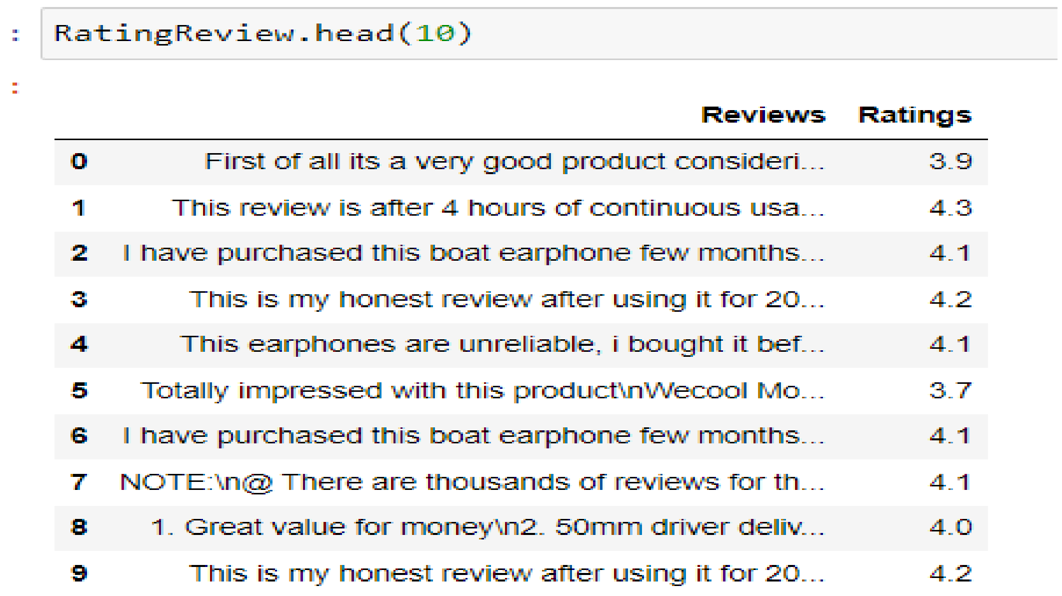
ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modeling of the Problem

Linear Algebra and Calculus concepts are used in the machine learning models. Since various classifiers have been used in the Machine Learning Algorithms, mathematics is working behind them. Calculation of evaluation metrics also involved mathematical concepts like algebraic summation. Use of Linear algebra in calculating the Euclidean distance in KNeighbors classifier. Log usage in DecisionTreeClassifier for calculating Information gain. In the statistical part descriptive statistics have been used to describe the data and to calculate various statistical parameters like mean, minimum value, maximum value, median value, count, standard deviation etc. And in the analytics modelling various techniques have been used like for analysing the data visualization countplot have been used.

Data Sources and their formats

Dataset that is used is having csv format of file and that is webscrapped from various sources like e-commerce websites amazon, flipkart etc.:



```
: RatingReview.head(10)
:
```

	Reviews	Ratings
0	First of all its a very good product consideri...	3.9
1	This review is after 4 hours of continuous usa...	4.3
2	I have purchased this boat earphone few months...	4.1
3	This is my honest review after using it for 20...	4.2
4	This earphones are unreliable, i bought it bef...	4.1
5	Totally impressed with this product\nWecool Mo...	3.7
6	I have purchased this boat earphone few months...	4.1
7	NOTE:\n@ There are thousands of reviews for th...	4.1
8	1. Great value for money\n2. 50mm driver deliv...	4.0
9	This is my honest review after using it for 20...	4.2

Text Pre-processing Done

In the text pre-processing and cleaning. Data have been checked for datatype and then various techniques have been used to clean the data like stopword removal, punctuation removal, lowercase, removal of frequent words, removal of rare words, Lemmatization, Emoji’s removal, URL removal, HTML tags removal.

Data Inputs- Logic- Output Relationships

No data-input-logic-output relationship has been found.

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

No assumption taken during model building.

Hardware and Software Requirements and Tools Used

In the hardware a laptop has been used along with an optical mouse.

In software excel, Python Jupyter notebook, have been used. Here is the table with list of libraries, import method and application of that method/function.

Library	**Import method/function**	**Application**
NumPy	array	Creating an array
pandas	DataFrame	importing DataFrame and other DataFrame related operations
matplolib, seaborn	countplot, distplot, scatterplots, pairplots	for data visualization
nltk.corpus	stopwords	For listing out the stopwords in a text.
	wordnet	For finding the meaning of words, synonym and antonym.
nltk.stem	WordNetLemmatizer	Used for lemmatization of a text.
sklearn.metrics	accuracy_score, confusion_matrix, classification_report	for calculating accuracy score for various classification model, making a classification report and creating confusion matrix for every classification model.
sklearn.ensemble	RandomForestClassifier, GradientBoostingClassifier	For importing various classifiers for machine learning algorithms
sklearn.neighbors	KNeighborsClassifier	For importing the classifier.
sklearn.tree	DecisionTreeClassifier	For importing the classifier.
sklearn.model_selection	train_test_split	for splitting dataset into training and testing data
sklearn	model_selection	For model selection.
	preprocessing	used for preprocessing the data.
	Linear model	Importing model.
	Naïve_bayes	Importing classifier.
	metrics	Importing metrics
	svm	Importing classifier
	decomposition	For decomposing the data.
	ensemble	For calling ensemble.
sklearn.feature_extraction.text	TfidfVectorizer	For converting the text to tfidf vectors.
	CountVectorizer	For converting the text to count vectors.
keras.preprocessing.text	Tokenizer	For tokenizing the text data.
keras.preprocessing.sequence	pad_sequences	
keras.preprocessing	text, sequence	For importing text and sequence.
keras.models	Model	For importing the model
	Input	For importing the input function

	Dense	For importing dense layer in model.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Problem solving approach includes text cleaning using various methods like stopword removal, punctuation marks removal, url removal, html tags removal, most frequent words removal, rare words removal, emoji's removal etc. Before that the average count of words for each class was calculated and visualization was done using seaborn plot. In machine learning various tokenized data have been used.

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

Algorithms used in the training and testing are:

RandomForestClassifier

SupportVectorClassifierr

GradientBoostingClassifier

KNeighborsClassifier

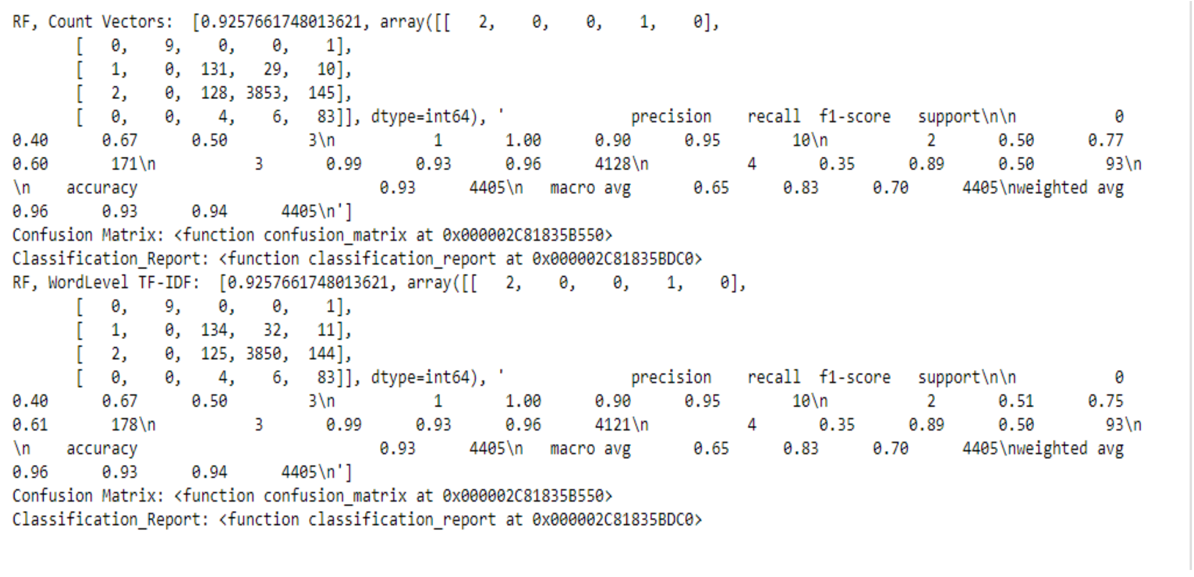
DecisionTreeClassifier

NaiveBayes Classifier

Run and Evaluate selected models

RandomForestClassifier is the first algorithm used for the classification which is a type of ensemble technique which is generated using a random selection of attributes at each node to determine the split. During classification each tree votes and the most popular class is returned. It is comparable to adaboost in accuracy but more robust to errors and outliers. This technique is insensitive to the number of attributes selected for consideration at each split, and faster than bagging and boosting.

Evaluation metrics and classification report for the algorithm is shown below.



```
RF, Count Vectors:  [0.9257661748013621, array([[   2,    0,    0,    1,    0],
       [   0,    9,    0,    0,    1],
       [   1,    0,  131,   29,   10],
       [   2,    0,  128, 3853,  145],
       [   0,    0,    4,    6,   83]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.67      0.50         3\n           1       1.00      0.90      0.95        10\n           2       0.50      0.77
0.60       171\n           3       0.99      0.93      0.96      4128\n           4       0.35      0.89      0.50        93\n
\n    accuracy                           0.93      4405\n   macro avg       0.65      0.83      0.70      4405\nweighted avg
0.96      0.93      0.94      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
RF, WordLevel TF-IDF:  [0.9257661748013621, array([[   2,    0,    0,    1,    0],
       [   0,    9,    0,    0,    1],
       [   1,    0,  134,   32,   11],
       [   2,    0,  125, 3850,  144],
       [   0,    0,    4,    6,   83]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.67      0.50         3\n           1       1.00      0.90      0.95        10\n           2       0.51      0.75
0.61       178\n           3       0.99      0.93      0.96      4121\n           4       0.35      0.89      0.50        93\n
\n    accuracy                           0.93      4405\n   macro avg       0.65      0.83      0.70      4405\nweighted avg
0.96      0.93      0.94      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
```

The second algorithm that has been used is **Support Vector Classifier.**

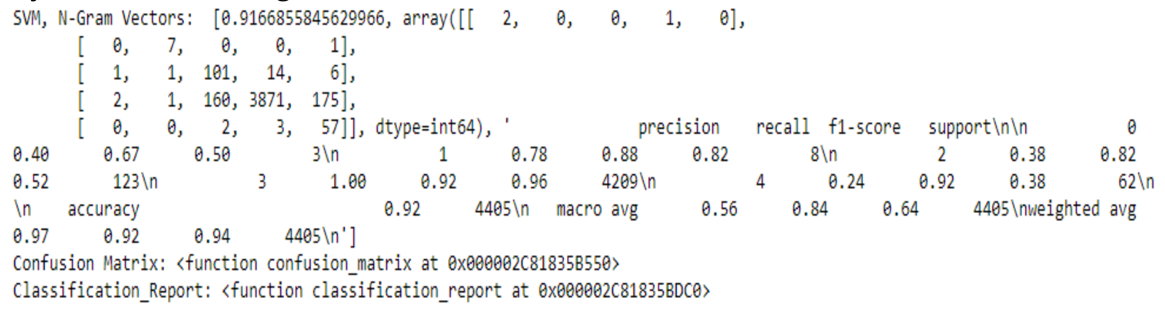
This algorithm is used for both categorical and continuous type of data. It constructs a hyperplane in multidimensional manner in an iterative manner, which is used to minimize the error. The core idea of the SVM is to create a hyperplane that best divides the dataset into classes. A hyperplane is a decision plane which separates between a set of objects having different class memberships. The main objective of the SVM is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. SVM searches for the maximum marginal hyperplane in following steps:

Generate hyperplanes which segregates the classes in the best way.

Select the right hyperplane with the maximum segregation from the nearest data points.

SVM is implemented in practice using a kernel. A kernel transforms an input data space into the required form. It converts non-separable problem to separable problems by adding more dimension to either nearest point is more useful in non-linear separation.

Key Metrics for the algorithm are as follows:



```
SVM, N-Gram Vectors:  [0.9166855845629966, array([[   2,    0,    0,    1,    0],
       [   0,    7,    0,    0,    1],
       [   1,    1,  101,   14,    6],
       [   2,    1,  160, 3871,  175],
       [   0,    0,    2,    3,   57]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.67      0.50         3\n           1       0.78      0.88      0.82         8\n           2       0.38      0.82
0.52       123\n           3       1.00      0.92      0.96      4209\n           4       0.24      0.92      0.38        62\n
\n    accuracy                           0.92      4405\n   macro avg       0.56      0.84      0.64      4405\nweighted avg
0.97      0.92      0.94      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
```

The third algorithm used is **GradientBoostingClassifier** which is an ensemble technique in which tries to build a weighted sum of weak learners. Mathematics behind this algorithm is:

$$S_L(.) = \sum_{l=1}^{L} C_l * W_l(.)$$

$$where\ C_l' sare\ coefficients\ and\ W_l's\ are\ the\ weak\ learners.$$

Gradient boosting cast the problem into gradient descent one. It also uses an iterative approach to find the optimal model. At each iteration weak learners are fit to the opposite of the gradient of the current fitting error with respect to the current ensemble model. First the theoretical gradient descent process over the ensemble model can be written as $S_l(.) = S_{l-1}(.) - C_l * \nabla_{sl-1} * E(S_{l-1})(.)\quad where\ E(.) is\ the\ fitting\ error\ of\ the\ given\ model, C_l\ is\ a\ coefficient\ corresponding\ to\ the\ step\ size\ and\ \nabla_{sl-1} * E(S_{l-1})(.)\quad is\ the\ opposite\ of\ the\ gradient\ of\ the\ fitting\ error\ with\ respect\ to\ the\ ensemble\ model\ at\ step\ l-1.$

This opposite of the gradient is a function that can, in the training dataset (for which we know inputs and outputs) these evaluations are called pseudo-residuals attached to each observation. Since these pseudo-residuals should not be added in the ensemble model only new instance weak model is to be added.

The following steps are followed in the algorithm:

Fit the best possible weak learner to pseudo-residuals.

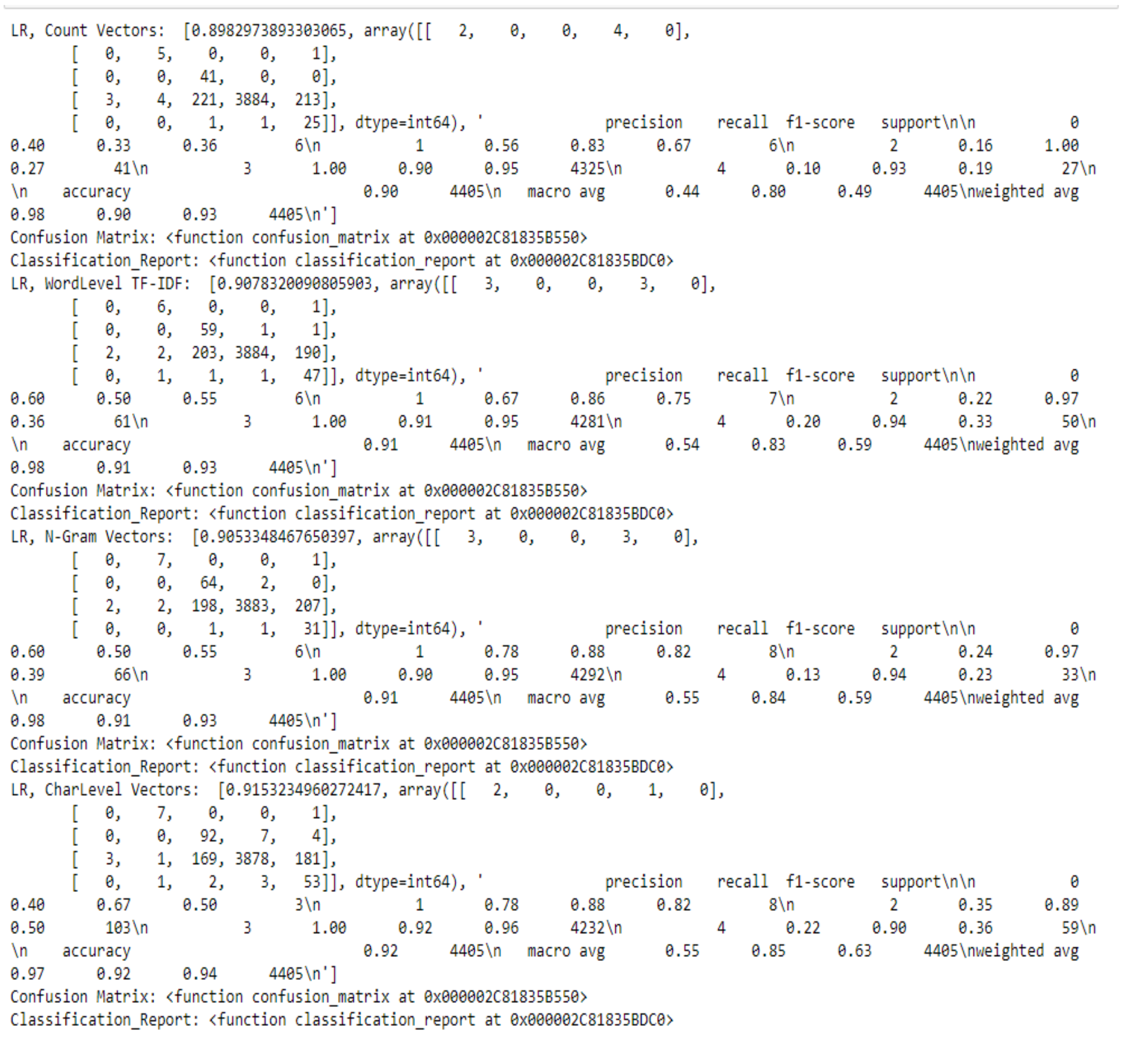
Compute the value of the optimal step-size that defines by how much we update the ensemble model in the direction of new weak learner

Update the ensemble model by adding the new weak learner multiplied by the step size.

Compute new pseudo-residuals that indicate for each observation, in which direction, we would like to update the next ensemble model predictions.

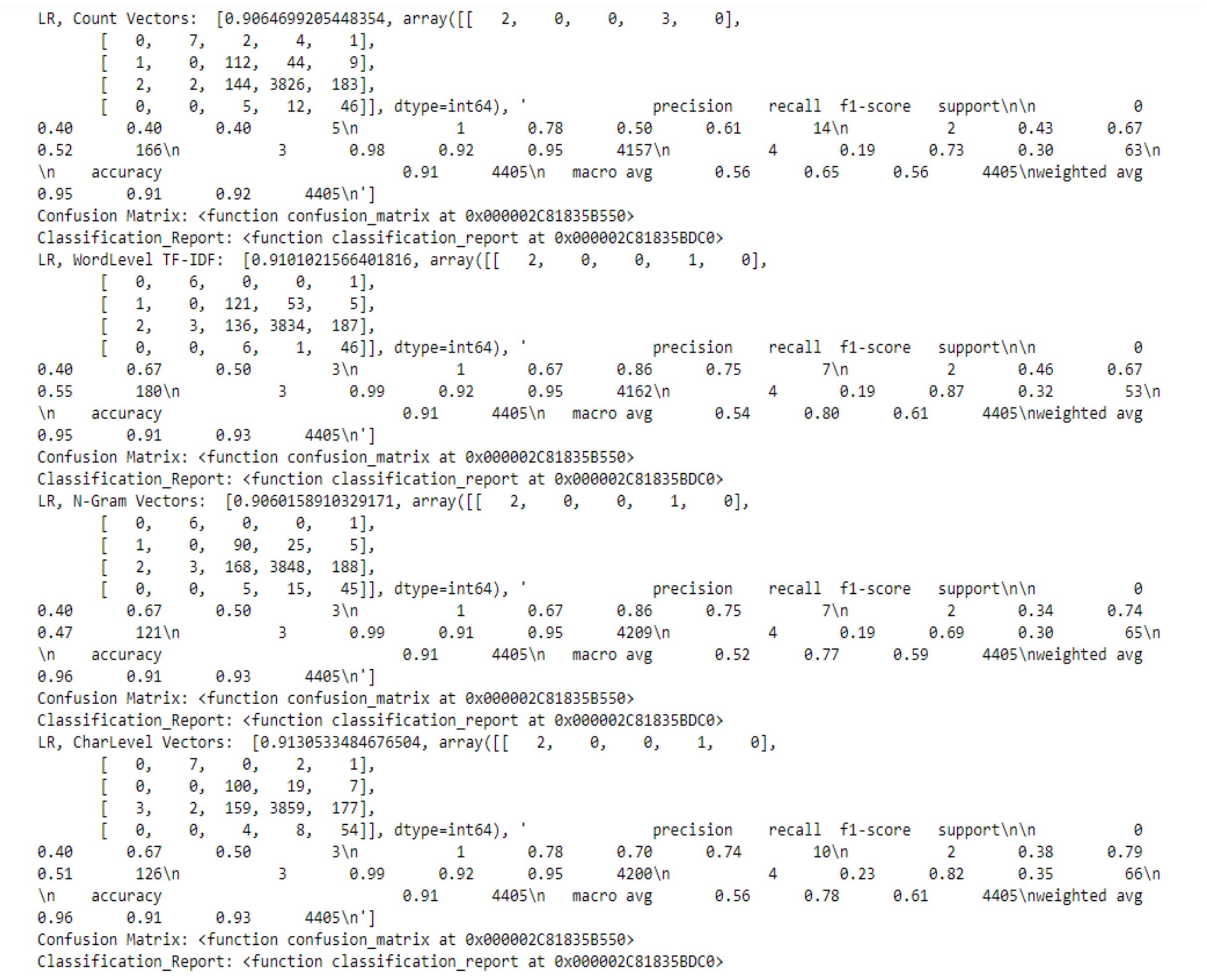
Repeating these steps, we build our sequentially L models and aggregate them following a gradient descent approach. Gradient boosting uses a gradient descent approach and can more easily be adapted to large number of loss function. Gradient boosting updates values of the observations at each iteration. Weak learners are trained to fit the pseudo-residuals that indicate in which direction to correct the current ensemble model predictions to lower the error. Following is the code of the algorithm:

The key metrics for the algorithm are as follows:



```
LR, Count Vectors:  [0.8982973893303065, array([[   2,    0,    0,    4,    0],
       [   0,    5,    0,    0,    1],
       [   0,    0,   41,    0,    0],
       [   3,    4,  221, 3884,  213],
       [   0,    0,    1,    1,   25]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.33      0.36         6\n           1       0.56      0.83      0.67         6\n           2       0.16      1.00
0.27        41\n           3       1.00      0.90      0.95      4325\n           4       0.10      0.93      0.19        27\n
\n    accuracy                           0.90      4405\n   macro avg       0.44      0.80      0.49      4405\nweighted avg
0.98      0.90      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, WordLevel TF-IDF:  [0.9078320090805903, array([[   3,    0,    0,    3,    0],
       [   0,    6,    0,    0,    1],
       [   0,    0,   59,    1,    1],
       [   2,    2,  203, 3884,  190],
       [   0,    1,    1,    1,   47]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.60      0.50      0.55         6\n           1       0.67      0.86      0.75         7\n           2       0.22      0.97
0.36        61\n           3       1.00      0.91      0.95      4281\n           4       0.20      0.94      0.33        50\n
\n    accuracy                           0.91      4405\n   macro avg       0.54      0.83      0.59      4405\nweighted avg
0.98      0.91      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, N-Gram Vectors:  [0.9053348467650397, array([[   3,    0,    0,    3,    0],
       [   0,    7,    0,    0,    1],
       [   0,    0,   64,    2,    0],
       [   2,    2,  198, 3883,  207],
       [   0,    0,    1,    1,   31]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.60      0.50      0.55         6\n           1       0.78      0.88      0.82         8\n           2       0.24      0.97
0.39        66\n           3       1.00      0.90      0.95      4292\n           4       0.13      0.94      0.23        33\n
\n    accuracy                           0.91      4405\n   macro avg       0.55      0.84      0.59      4405\nweighted avg
0.98      0.91      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, CharLevel Vectors:  [0.9153234960272417, array([[   2,    0,    0,    1,    0],
       [   0,    7,    0,    0,    1],
       [   0,    0,   92,    7,    4],
       [   3,    1,  169, 3878,  181],
       [   0,    1,    2,    3,   53]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.67      0.50         3\n           1       0.78      0.88      0.82         8\n           2       0.35      0.89
0.50       103\n           3       1.00      0.92      0.96      4232\n           4       0.22      0.90      0.36        59\n
\n    accuracy                           0.92      4405\n   macro avg       0.55      0.85      0.63      4405\nweighted avg
0.97      0.92      0.94      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
```

The fourth algorithm used is the **KNeighborsClassification**, it is very simple to understand versatile and one of the topmost machine learning algorithms. In KNeighborsClassification is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When k=1 then algorithm is known as the nearest neighbour algorithm. Suppose P1 is the point, for which label needs to predict. First, you find the k-closest point to P1 and then classify points by majority vote of its K-neighbors. Each object votes for their class and the class with the most votes are taken as the prediction. For finding the closest similar points, we need to find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNeighborsClassifier has the following basic steps:
Calculate the distance
Find the closest neighbors
Vote for labels.
KNeighborsClassifier performs best with the lower number of features than large number of features.
Evaluation metrics and classification report for the algorithm is shown below:



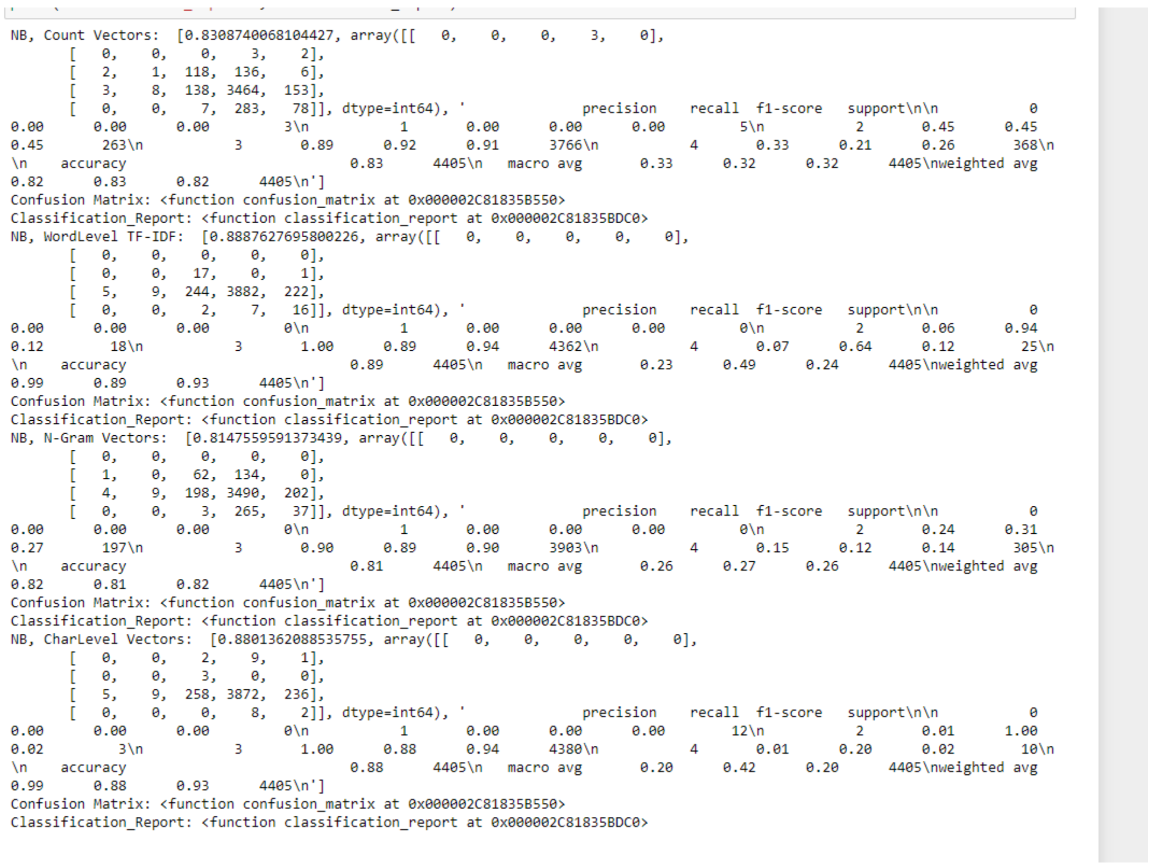
```
LR, Count Vectors:  [0.9064699205448354, array([[   2,    0,    0,    3,    0],
       [   0,    7,    2,    4,    1],
       [   1,    0,  112,   44,    9],
       [   2,    2,  144, 3826,  183],
       [   0,    0,    5,   12,   46]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.40      0.40         5\n           1       0.78      0.50      0.61        14\n           2       0.43      0.67
0.52       166\n           3       0.98      0.92      0.95      4157\n           4       0.19      0.73      0.30        63\n
\n    accuracy                           0.91      4405\n   macro avg       0.56      0.65      0.56      4405\nweighted avg
0.95      0.91      0.92      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, WordLevel TF-IDF:  [0.9101021566401816, array([[   2,    0,    0,    1,    0],
       [   0,    6,    0,    0,    1],
       [   1,    0,  121,   53,    5],
       [   2,    3,  136, 3834,  187],
       [   0,    0,    6,    1,   46]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.67      0.50         3\n           1       0.67      0.86      0.75         7\n           2       0.46      0.67
0.55       180\n           3       0.99      0.92      0.95      4162\n           4       0.19      0.87      0.32        53\n
\n    accuracy                           0.91      4405\n   macro avg       0.54      0.80      0.61      4405\nweighted avg
0.95      0.91      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, N-Gram Vectors:  [0.9060158910329171, array([[   2,    0,    0,    1,    0],
       [   0,    6,    0,    0,    1],
       [   1,    0,   90,   25,    5],
       [   2,    3,  168, 3848,  188],
       [   0,    0,    5,   15,   45]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.67      0.50         3\n           1       0.67      0.86      0.75         7\n           2       0.34      0.74
0.47       121\n           3       0.99      0.91      0.95      4209\n           4       0.19      0.69      0.30        65\n
\n    accuracy                           0.91      4405\n   macro avg       0.52      0.77      0.59      4405\nweighted avg
0.96      0.91      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, CharLevel Vectors:  [0.9130533484676504, array([[   2,    0,    0,    1,    0],
       [   0,    7,    0,    2,    1],
       [   0,    0,  100,   19,    7],
       [   3,    2,  159, 3859,  177],
       [   0,    0,    4,    8,   54]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.67      0.50         3\n           1       0.78      0.70      0.74        10\n           2       0.38      0.79
0.51       126\n           3       0.99      0.92      0.95      4200\n           4       0.23      0.82      0.35        66\n
\n    accuracy                           0.91      4405\n   macro avg       0.56      0.78      0.61      4405\nweighted avg
0.96      0.91      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
```

NaiveBayesClassifier algorithm is based on bayes theorem with the assumption that all the features that predicts the target value are independent of each other. It works well with the NLP problems. Naives bayes classifier works on the conditional probability given by the formula: $P(C/X) = \frac{P(X/C)*P(C)}{P(X)}$

$$P\,(C/X) = P(X_1/C) * P(X_2/C) * \ldots \ldots * P(X_n/C) * P(C)$$

In this classifier the prior probability is first calculated than the posteriori probability is calculated. In the prior probability of hypothesis H, the initial probability before we observe any data, is calculated then probability of sample data is observed. And finally, probability of observing the sample X, given that the hypothesis holds. Its advantages include easy implementation, good results obtained in most of the results. And disadvantages are loss of accuracy due to class conditional independence., computational cost is high.

Metrics for Naïve Bayes are as follows:



```
NB, Count Vectors:  [0.8308740068104427, array([[   0,    0,    0,    3,    0],
       [   0,    0,    0,    3,    2],
       [   2,    1,  118,  136,    6],
       [   3,    8,  138, 3464,  153],
       [   0,    0,    7,  283,   78]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.00      0.00      0.00         3\n           1       0.00      0.00      0.00         5\n           2       0.45      0.45
0.45       263\n           3       0.89      0.92      0.91      3766\n           4       0.33      0.21      0.26       368\n
\n    accuracy                           0.83      4405\n   macro avg       0.33      0.32      0.32      4405\nweighted avg
0.82      0.83      0.82      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
NB, WordLevel TF-IDF:  [0.8887627695800226, array([[   0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0],
       [   0,    0,   17,    0,    1],
       [   5,    9,  244, 3882,  222],
       [   0,    0,    2,    7,   16]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.00      0.00      0.00         0\n           1       0.00      0.00      0.00         0\n           2       0.06      0.94
0.12        18\n           3       1.00      0.89      0.94      4362\n           4       0.07      0.64      0.12        25\n
\n    accuracy                           0.89      4405\n   macro avg       0.23      0.49      0.24      4405\nweighted avg
0.99      0.89      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
NB, N-Gram Vectors:  [0.8147559591373439, array([[   0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0],
       [   1,    0,   62,  134,    0],
       [   4,    9,  198, 3490,  202],
       [   0,    0,    3,  265,   37]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.00      0.00      0.00         0\n           1       0.00      0.00      0.00         0\n           2       0.24      0.31
0.27       197\n           3       0.90      0.89      0.90      3903\n           4       0.15      0.12      0.14       305\n
\n    accuracy                           0.81      4405\n   macro avg       0.26      0.27      0.26      4405\nweighted avg
0.82      0.81      0.82      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
NB, CharLevel Vectors:  [0.8801362088535755, array([[   0,    0,    0,    0,    0],
       [   0,    0,    2,    9,    1],
       [   0,    0,    3,    0,    0],
       [   5,    9,  258, 3872,  236],
       [   0,    0,    0,    8,    2]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.00      0.00      0.00         0\n           1       0.00      0.00      0.00        12\n           2       0.01      1.00
0.02         3\n           3       1.00      0.88      0.94      4380\n           4       0.01      0.20      0.02        10\n
\n    accuracy                           0.88      4405\n   macro avg       0.20      0.42      0.20      4405\nweighted avg
0.99      0.88      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
```

Last Classifier is **DecisionTreeClassifier** which consists of leaf, nodes and roots for making the decision. It has an inverted tree like structure. These decision trees are well known for their capability to capture the patterns in the data. The main parts of the decisiontree includes Gini impurity, entropy and information gain. Entropy is the amount of information that is needed accurately to describe some sample. It is given by formula:

$$Entropy = -\sum_{i=1}^{n} p_i * \log(p_i).$$

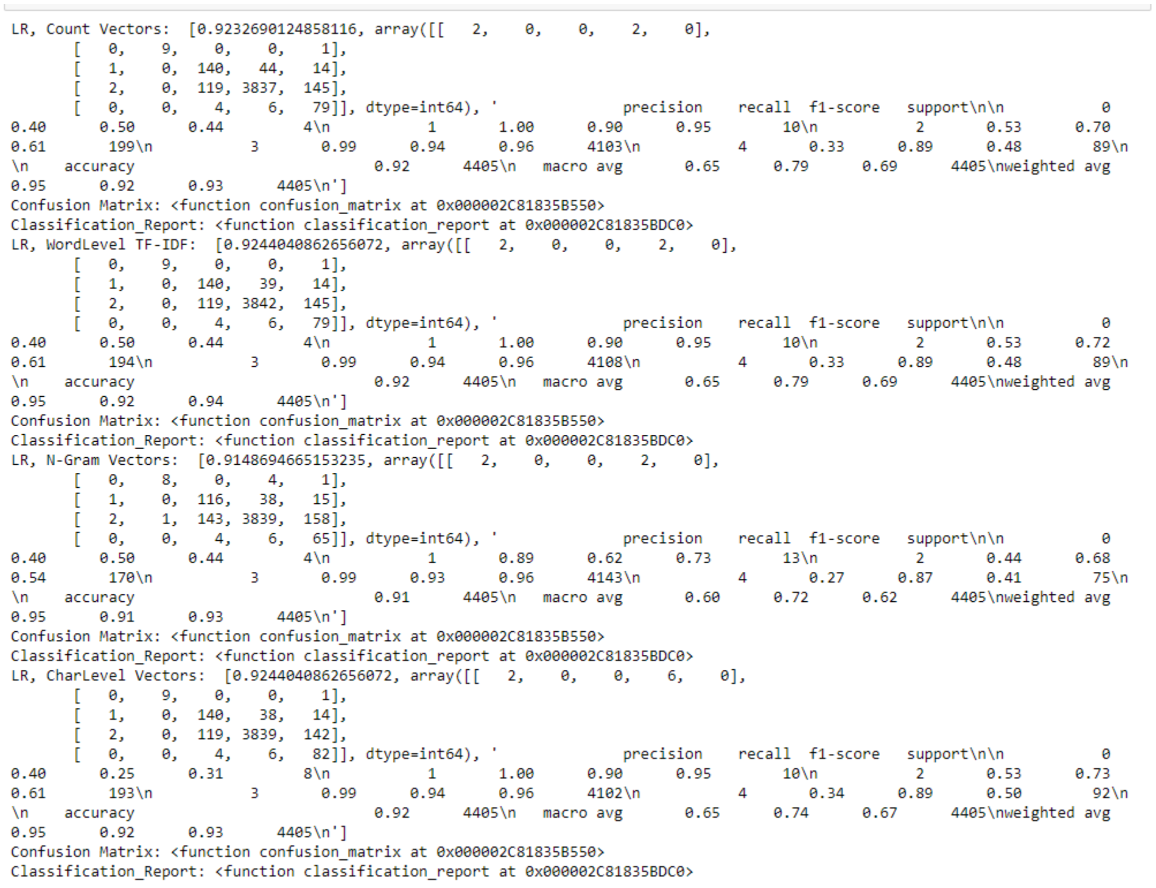
The other part is Gini impurity which is given by the formula

$$Gini\ index = 1 - \sum_{i=1}^{n} p_i^2$$

The third part is Information Gain which is defined as the information gained by any attribute. It is given by the formula:

$$Gain(A) = Info(D) - Info_A(D)$$

Key Metrics for DecisionTreeClassifier are as follows:



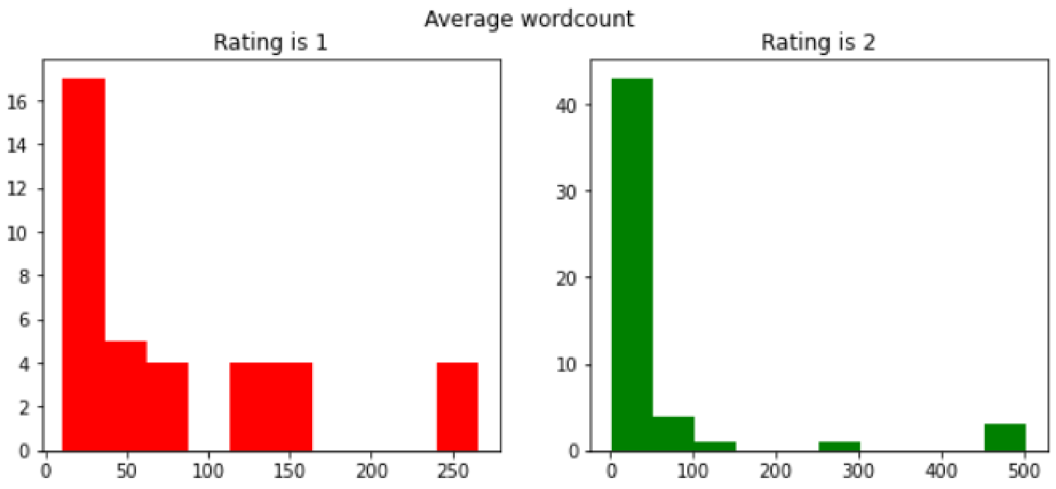
```
LR, Count Vectors:  [0.9232690124858116, array([[   2,    0,    0,    2,    0],
       [   0,    9,    0,    0,    1],
       [   1,    0,  140,   44,   14],
       [   2,    0,  119, 3837,  145],
       [   0,    0,    4,    6,   79]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.50      0.44         4\n           1       1.00      0.90      0.95        10\n           2       0.53      0.70
0.61       199\n           3       0.99      0.94      0.96      4103\n           4       0.33      0.89      0.48        89\n
\n    accuracy                           0.92      4405\n   macro avg       0.65      0.79      0.69      4405\nweighted avg
0.95      0.92      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, WordLevel TF-IDF:  [0.9244040862656072, array([[   2,    0,    0,    2,    0],
       [   0,    9,    0,    0,    1],
       [   1,    0,  140,   39,   14],
       [   2,    0,  119, 3842,  145],
       [   0,    0,    4,    6,   79]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.50      0.44         4\n           1       1.00      0.90      0.95        10\n           2       0.53      0.72
0.61       194\n           3       0.99      0.94      0.96      4108\n           4       0.33      0.89      0.48        89\n
\n    accuracy                           0.92      4405\n   macro avg       0.65      0.79      0.69      4405\nweighted avg
0.95      0.92      0.94      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, N-Gram Vectors:  [0.9148694665153235, array([[   2,    0,    0,    2,    0],
       [   0,    8,    0,    4,    1],
       [   1,    0,  116,   38,   15],
       [   2,    1,  143, 3839,  158],
       [   0,    0,    4,    6,   65]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.50      0.44         4\n           1       0.89      0.62      0.73        13\n           2       0.44      0.68
0.54       170\n           3       0.99      0.93      0.96      4143\n           4       0.27      0.87      0.41        75\n
\n    accuracy                           0.91      4405\n   macro avg       0.60      0.72      0.62      4405\nweighted avg
0.95      0.91      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
LR, CharLevel Vectors:  [0.9244040862656072, array([[   2,    0,    0,    6,    0],
       [   0,    9,    0,    0,    1],
       [   1,    0,  140,   38,   14],
       [   2,    0,  119, 3839,  142],
       [   0,    0,    4,    6,   82]], dtype=int64), '              precision    recall  f1-score   support\n\n           0
0.40      0.25      0.31         8\n           1       1.00      0.90      0.95        10\n           2       0.53      0.73
0.61       193\n           3       0.99      0.94      0.96      4102\n           4       0.34      0.89      0.50        92\n
\n    accuracy                           0.92      4405\n   macro avg       0.65      0.74      0.67      4405\nweighted avg
0.95      0.92      0.93      4405\n']
Confusion Matrix: <function confusion_matrix at 0x000002C81835B550>
Classification_Report: <function classification_report at 0x000002C81835BDC0>
```

Key Metrics for success in solving problem under consideration

Key Metrics for success in solving problem under consideration includes Accuracy. Accuracy for different Classification Algorithm has been found out at count vector level, Word-level, N-gram vectors level and charlevel vectors. Confusion Matrix and Classification Report has been printed and precision, recall, f1-score has been found for each class. All the values of confusion matrix like TP, FP, TN, FN has been found for measuring the performance of the model.

Visualizations

Visualization plots that have been used in the project includes histplots for visualizing the Ratings of various parameters for char_count, word_count. To see the see the average word_count and average char_count for various ratings from Rating 1,2,3,4,5 plots at different Rating have been found.



Interpretation of the Results

From the visualization we can say that the average wordcount for rating 3 is least whereas highest for rating 1. Average character count for Rating 3 is found to be minimum and Rating 1 is found to be maximum. For Rating1 average wordcount histogram suggests that maximum value of Average wordcount lies between 0-50 and same trend is followed by all the Ratings except Rating4 and Rating 5 where Average wordcount lies between 0-100. From the average char count plot, we can say that maximum value lies between 0-200 for Rating 1 and for Rating 2 we have average char count is 0-250. For rating 3 we have maximum average char_count between 0-500. For Rating 4 we have maximum average char_count between 0-1000. And for Rating 5 we have maximum average char_count between 0-300.

CONCLUSION

Key Findings and Conclusions of the Study

From the complete project we came to know that starting from importing the dataset and to the end various techniques have been used from text visualization to using various text pre-processing techniques like stopwords removal, punctuation removal, lowercase, removal of frequent words, removal of rare words, lemmatization, emoji's removal, URL removal, html tags removal. Then feature engineering techniques like count-vectors, TF-IDF vectors, NLP based features like 'char_count','word_count','word_density', 'punctuation_count','title_word_count','upper_case_word_count' have been used for various machine learning models and various key_metrics have been found like accuracy, classification_report and confusion_matrix. And finally based on the accuracy xgboost have been selected as the best model having highest accuracy. This model can be used for predicting the further Ratings based on reviews.

Learning Outcomes of the Study in respect of Data Science

Learnings includes deep analysis of the text data. In text data visualization of word_count, char_count and their mean value has been used which plays an important role in visualizing the mean word_count and mean char_count. And in the text-pre-processing various techniques like stopwords removal, punctuation removal, frequent words removal, rare words removal, URL removal, html tags removal has been done to clean the text data, this process plays an important role in building machine learning model because properly cleaned data will provide best data to be put in the machine learning model to produce accurate results. Then various machine learning models have been used for classification and various key metrics have been found like accuracy, confusion_matrix, classification_report to measure the performance of the model. And finally based on the results the xgboost have been selected as the best model.

Limitations of this work and Scope for Future Work

Limitations of this work is that more robust model can be build using some more advanced techniques in machine learning model building and Exhaustive EDA can also improve the accuracy of the model using some more techniques for the EDA analysis can solve the problem of less accuracy. Future work can be done in the text pre-processing and data visualization part for analysing data more effectively. Neural Networks can also be used for classification of text.