Part 1: Text Mining

Data Pre-processing

def normalizer(): This is user defined function used for cleansing the text data.

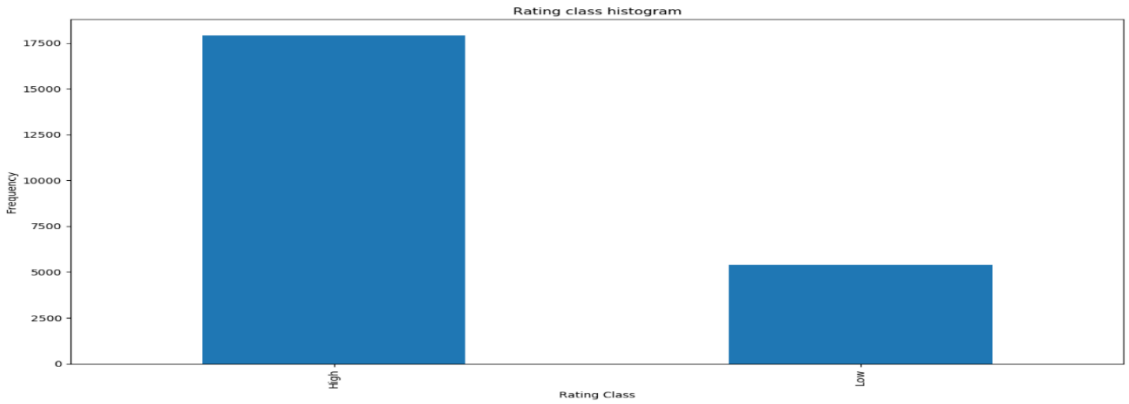
Using 'BeautifulSoup' function from 'bs4' library and 'lxml' toolkit to remove XML and HTML encoding from tweets which come due to some hyperlink present in tweet. Also, Regular Expression(Regrex) is used to remove any numbers hyperlinks hastags, etc this expression is implemented using manual scripts. Expression used in this code is: `"(@[A-Za-z0-9]+)|([^A-Za-z \t])|(\w+:\/\/\S+)"`

As we cannot explicitly implement machine learning model on raw text, we need to split each word like this `['one', 'morning,', 'when', 'gregor', 'samsa', 'woke', 'from'].` And convert all the words in lowercase as case sensitivity changes the impact of same word for machine.

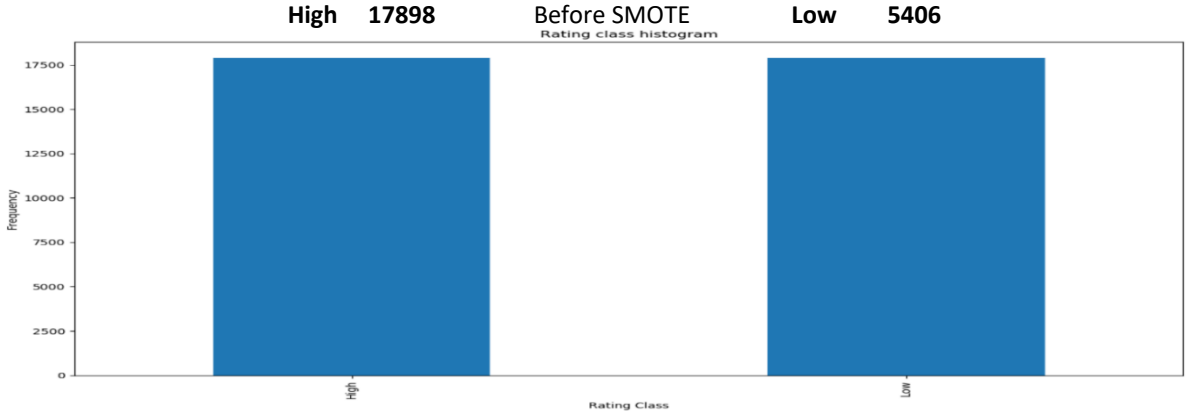
To remove stop words, like to, and, the, etc. from the tweet I have implemented stopword() function from nltk.corprus librar. We need to mention 'english' as attribute to remove stop words of English language there are 127 stop words for English language in NLTK.

Lemmatizing words having similar meaning despite its tense using WordNet Corpus. e.g. Was -> (to) be, better -> good, cats -> cat keeping the basic meaning of the word.

Data Balancing: SMOTE(Synthetic Minority Over-sampling Technique) is used as the graph shows data is highly imbalanced and hence to avoid model taking bias decision which can affect the overall output of the deployed model.



High 17898 Before SMOTE **Low 5406**



High 17898 After Smote **Low 17898**

`TfidfVectorizer ():` It is a function from sklearn library under feature_extraction.text class it takes several attributes like 1) min_df – When making the vocabulary, it ignores strictly terms that have documented for less than given value having default value = 1. In this code min_df is set as 5. 2) ngram_range – It extracts the words as a single word based on the lower and upper boundary of the range of n-values. Which is (1, 2) for this code hence including unigrams and bigrams. `tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', ngram_range=(1, 2))` this are parameters for model creation code.

Data Interpretation

Cross-Validation Score and Confusion Matrix for Support Vector Classifier and Naïve Bayes Classifier where n_splits for KFold is 10 and shuffle=True and balanced target variable Y_SMOTE is used.

Support Vector Classifier:

Mean CV accuracy SVC	Best CV accuracy for SVC	TP	TN	FP	FN
0.9107879430626212	**0.9201716738197425**	422	1722	55	131

Naïve Bayes Classifier

Mean CV accuracy for NBC	Best CV accuracy for NBC	TP	TN	FP	FN
0.8327322356077721	**0.8434148434148434**	183	1783	13	352

Conclusion

Based on the outcomes of the number it tells that SVC is performing well as compared to NBC on training set MedReviews.csv. Best Cross-Validation accuracy for SVC is 92% making difference of 8% from that of NBC. Also, average mean Cross-Validation value for SVC is higher than NBC by 8%. Based on this, the SVC model is saved, and the vocabulary file is generated using this knowledge about the data for deployment process. After performing data normalization and cleansing SVC model is deployed on NoRatings.csv which will be test set. The vocabulary file generated from model creation code and SVC file which is created when model is saved will be implemented during deployment process. Therefore, after deployment new CSV file will be saved which will have predicted target class as per the SVC model.

Part 2: Decision Tree Ensembles

- ### Making Dataset adaptive for modelling:

- Handling categorical variables: Transaction and Weekend class is converted using defined function 'converter'. Month and VisitorType class are converted into numeric using the inbuilt map function.
- Dividing dataset: The numerically converted dataset is then divided into label and feature sets where X: Independent variables and Y: Target variable (Transaction).
- Normalization: Once all the values of each class are converted into numeric it is necessary to normalize it so that each feature has Mean as 0 and Variance as 1. Done using StandardScaler function from sklearn.
- Splitting: Dataset is divided into train and test for proportion of 70% and 30% respectively for all three models and setting seed as 9 throughout the code to get same output each time it is executed.
- Balancing: As shown by the plot, data is highly imbalance hence to avoid its consequence on the training model SMOTE (Synthetic Minority Over-sampling Technique) is used to balance the target variable.

Once pre-processing steps on dataset is performed its necessary to tune the models.

- ### Justification for hyperparameter tuning

GridSearchCV and RandomizedSearchCV is useful in finding the best parameter values for different model instead of manually performing we just need to provide collection of parameters and its values and in return it will provide best fit for the parameters on applied model. Therefore, justification for setting Recall as Scoring Parameter while using GridSearchCV and RandomizedSearchCV for each model:

As per the confusion matrix:

False Positive: Where Model predicts True for transaction class but, its False hence just loss of provided service not loss of valuable customer.

False Negative: Where model predicts False for transaction class but, its True hence loosing customer who would have shopped.

For any e-commerce website it's important to know how many customers are using it and number of transaction taking place. As this will help in gaining some business intelligence by predicting the growth or knowing the trend, etc. Hence it is essential that developed model predicts more of the ---

True Transaction class rather than False Transaction class. As per the definition recall is useful in finding the answer for How many relevant classes are predicted true? As it provides percentage for total relevant results correctly classified by your applied model. Therefore, while focusing on increasing number of transaction it is necessary to reduce False Negative predictions.

- ## Random Forest

Tuning without Feature Scaling

Best Parameters for Random Forest without feature scaling: {'n_estimators': 40, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 38, 'criterion': 'entropy', 'bootstrap': False}

Significant Values and Classification Metrics

Features	Significant values
PageValue	0.390358
Month	0.090054
ProductRelated_Duration	0.086705
Administrative	0.083305
ExitRate	0.083305
ProductRelated	0.081132
BounceRate	0.059919
Administrative_Duration	0.051204
Informational	0.022770
Informational_Duration	0.020775
VisitorType	0.013124
Weekend	0.010509
SpecialDay	0.006623

	Precision	Recall	F1-Score	Support
No Transaction	0.94	0.92	0.93	3091
Transaction	0.64	0.70	0.67	583
Accuracy			0.89	3674
Macro Avg.	0.79	0.81	0.80	3674
Weighted Avg.	0.89	0.89	0.89	3674

Tuning after Feature Scaling

Hence after observing the output of significant values of each feature I decided to remove least prominent features which are VisitorType, Weekend, SpecialDay the resultant values are better and hence avoided the trap of over-fitting the model.

Best Parameters for Feature Scaled Random Forest: {'n_estimators': 40, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 38, 'criterion': 'entropy', 'bootstrap': False}

Significant Values and Classification Metrics

Features	Significant Values
PageValue	0.405287
ExitRate	0.099144
Month	0.095384
ProductRelated_Duration	0.089888
ProductRelated	0.089181
Administrative	0.065328
BounceRate	0.061292
Administrative_Duration	0.055520
Informational_Duration	0.020901
Informational	0.018075

	Precision	Recall	F1-Score	Support
No Transaction	0.95	0.92	0.93	3091
Transaction	0.62	0.72	0.66	583
Accuracy			0.88	3674
Macro Avg.	0.78	0.82	0.80	3674
Weighted Avg.	0.89	0.88	0.89	3674

- ## AdaBoost Model

Hyperparameter Tuning

Best estimators for AdaBoostClassifier: (algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=9, random_state=9)

Significant Values and Classification Metrics

Features	Significant Values
Month	0.333333
PageValue	0.222222
Admiinistrative	0.222222
Visitor Type	0.111111
ExitRate	0.111111
Weekend	0.000000
SpecialDay	0.000000
BounceRate	0.000000
ProductRelated_Duration	0.000000
ProductRelated	0.000000
Informational_Duration	0.000000
Administrative_Duration	0.000000

	Precision	Recall	F1-Score	Support
No Transaction	0.96	0.89	0.92	3091
Transaction	0.57	0.78	0.66	583
Accuracy			0.87	3674
Macro Avg.	0.76	0.83	0.79	3674
Weighted Avg.	0.89	0.87	0.88	3674

- ## Gradient Boost

Hyperparameter tuning

Best estimators for Gradient Boost Classifier: {'n_estimators': 50, 'min_samples_leaf': 12, 'max_depth': 13}

Significant Values and Classification Metrics

Features	Significant Values
PageValue	0.634489
Month	0.086293
Administrative	0.056778
ProductRelated	0.048463
ExitRate	0.035557
ProductRelated_Duration	0.034863
Administrative_Duration	0.033281
BounceRate	0.029766
Informational_Duration	0.014315
Informational	0.012526
VisitorType	0.007269
Weelend	0.005227
SpecialDay	0.001173

	Precision	Recall	F1-Score	Support
No Transaction	0.95	0.93	0.94	3091
Transaction	0.65	0.72	0.68	583
Accuracy			0.89	3674
Macro Avg.	0.80	0.83	0.81	3674
Weighted Avg.	0.90	0.89	0.90	3674

Interpretation

Best Score for all the models is generated using GridSearchCV and RandomizedSearchCV were cross validation score is 5:

Model	Best Score
Random Forest without Feature Scaling	0.9533111885924249
Random Forest with Feature Scaling	0.9530356117010687
Ada Boosting	0.8818478534958544
Gradient Boosting	0.9349915345660825

Confusion Matrix and ROC_AUC_SCORE for all the models:

Model	True Positive	True Negative	False Positive	False Negative	ROC_AUC_SCORE
Random Forest without Feature Scaling	411	2855	236	172	0.8143117877221148
Random Forest with Feature Scaling	419	2831	260	164	0.8172906124292681
Ada Boosting	454	2743	348	129	0.8330728896430905
Gradient Boosting	422	2863	228	161	0.8250398295721602

Recommendations

After tuning hyperparameters for all the models, the insights from the numbers provided is valuable for justification of recommendation. AdaBoost fits in for the criteria as it provides least False Negative values with highest ROC_AUC_SCORE despite having low Best Score. Providing 0.8330728896430905 as ROC_AUC_SCORE. Also, it is less complex and does not need for feature scaling as its algorithm manages to provide best result by giving weightages to appropriate classes only. And compared to Random Forest and Gradient Boost its hyperparameter tuning and model deployment results are noticeable quicker.

Therefore, implementing AdaBoost for real-world solution would be the best choice amongst all to increase the transaction.