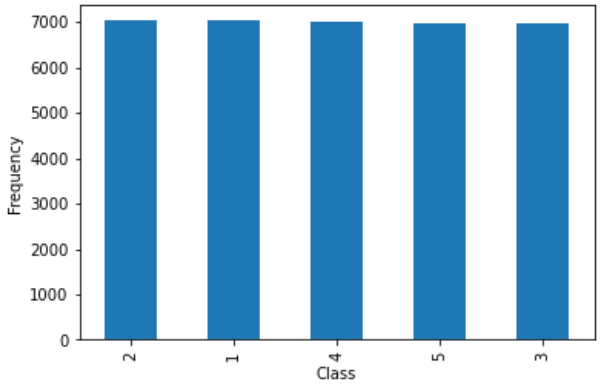
Sentiment Analysis on Hotel Reviews – A Multi Class Text Classification

Abstract:

In this small project, I am showing how to build multiclass text classification machine learning (ML) models to classify hotel reviews into a five-star scale. The goal of this project is to build ML classifiers that predict the rating of the reviews. Different classification models were trained on a big sample of hotel reviews. The reviews were preprocessed by removing punctuations and nonalphabetic characters and stop words. The features used in the final models were mainly n-grams of term frequency-inverse document frequency (tf-idf) of word lemmas. The best model with the highest performance was Logistic Regression with accuracy measures of 75%.

Datasets:

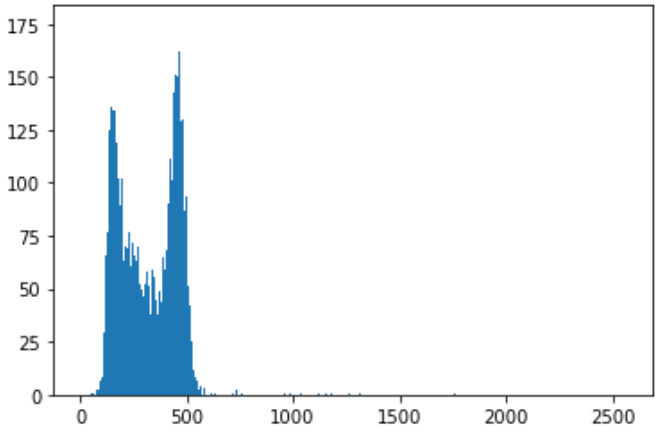
The training data set contains 35004 distinct reviews and the classes are almost evenly distributed. The following figure shows the distribution of the classes of the reviews within the training set.



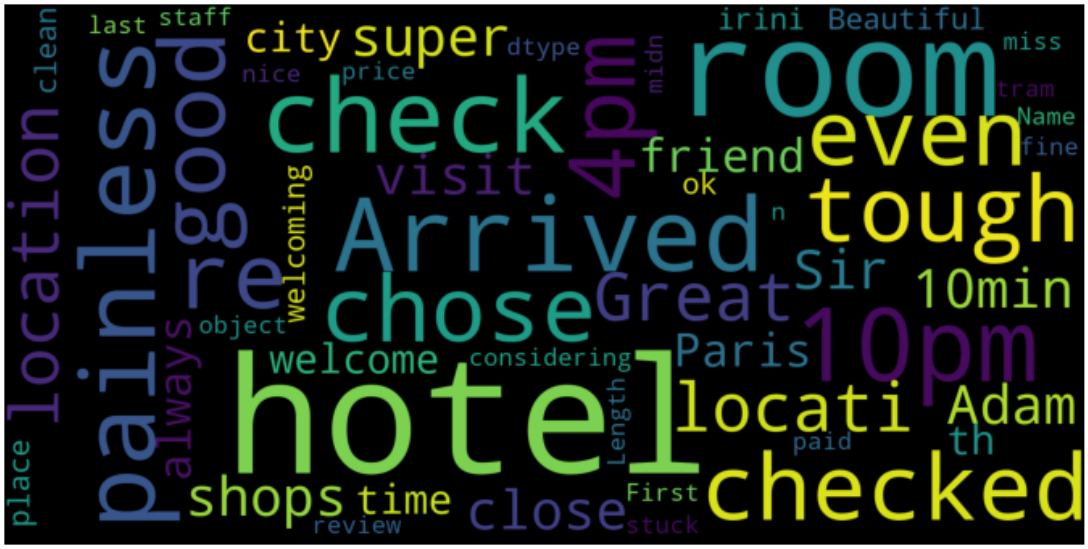
I also did some exploratory analyses on the reviews in terms of lengths and words they contain. The following are some statistics on the reviews' lengths in terms of the number of words. The next figure also shows the distribution of the text lengths of the reviews. The majority are within 500 words.

```
count    35004.000000
mean       321.172952
std        136.038517
min          4.000000
25%        192.000000
50%        321.000000
75%        444.000000

max       2561.000000
```



Next, I am showing the word cloud count on the entire corpus for the whole word sets. It shows the most frequent 80 words.



Preprocessing and Features:

For preprocessing, normalizations were applied. This includes reviews tokenization, lowercasing, removing stop words, special and non-alphabetical characters, in addition to stemming and lemmatization [8]. I tried both stemming and lemmatization separately, and applying lemmatization yielded better results, therefore, I ignored the stemming step (I came to a similar conclusion in a different published study [9] and this one also confirms my observation).

I noted that one row has a wrong value (text as a label), this was manually removed. If there were many, then this can be easily automated by deleting rows with NaNs and non-integer labels. The following picture shows the most 80 frequent normalized lemmas in the corpus after the preprocessing step.

After normalizing and preparing the reviews, I applied term frequency-inverse document frequency (tf-idf) to vectorize the reviews. The weights for the terms in the documents were used as features that were then fed to the classifiers. Note, the terms here are unigrams, bigrams, and trigrams of the lemmatized words in the reviews while the documents are the reviews.

The Models:

I started with a Decision Tree classifier as a baseline. I then trained four additional classifiers namely, Logistic Regression [4], Random Forest [5], XGBoost [6], and CatBoost [7]. These were trained using 5-fold cross-validation with different combinations of parameters in order to maximize the model performance. This was done using a grid search parameter tuning method. Class imbalance handling was not required in this task as the classes were already balanced. However, a function was added in case a newer dataset with imbalanced classes is to be trained. I also added a feature selection function. In this step, different numbers of features can be tried while monitoring the models' performance. However, I discarded this step as it was possible to select the top n features when vectorizing the reviews. To evaluate the models' performance, I used four metrics, namely, accuracy, precision, recall, and F1 measure. I also used the classification confusion matrix to show the classification distribution across the five classes.

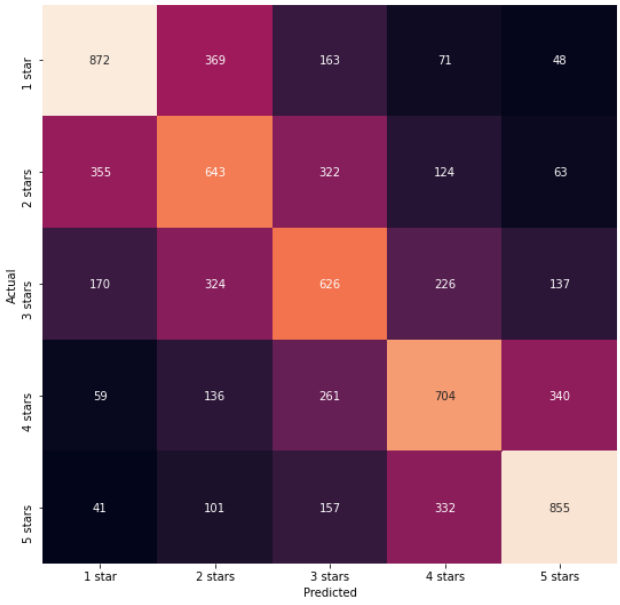
Results:

The performance of the used classifiers varied, and the best performance was obtained using the Logistic Regression classifier. In the next table, I am showing and comparing the performance of the different models according to the above-mentioned performance metrics. The code can be found in [3].

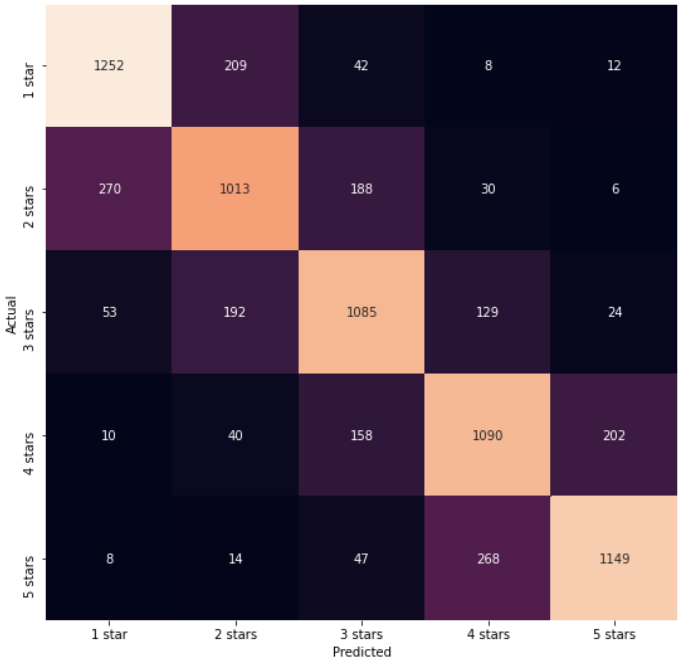
Classifier	Accuracy	Precision	Recall	F1
Decision Tree (baseline)	0.49	0.49	0.49	0.49
Logistic Regression	0.75	0.75	0.75	0.75
Random Forest	0.67	0.67	0.67	0.67
XGBoost	0.72	0.72	0.72	0.72
CatBoost	0.73	0.73	0.73	0.73

Below, I am showing the confusion matrix of the classifiers used.

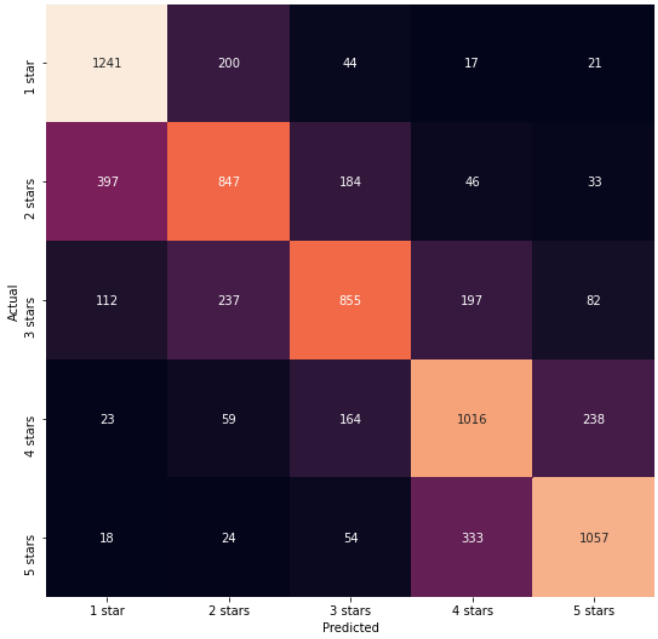
Baseline (Decision Tree)



Logistic Regression



Random Forest



Top features (visualization)

I used SHapley Additive exPlanation (SHAP) [11] to show the top features that the classifiers used when classifying the reviews.

Deployment:

I added a code to deploy the models (only best one) using Flask to make new predictions on new submitted reviews. You can check it out in the **passive_app.py** file. A test request is also added in the **request.py** file.

Discussion:

I believe the predictions make sense. By looking at the confusion matrix (take the logistic regression one for example) and misclassifications, most of the time, the majority of them were mistaken with either the upper-level or lower-level class. For example, the majority of 4 stars reviews were either mistaken with either the 3-star or the 5-star labels. This is due to the similarity in the language and vocabulary used when writing these reviews especially that the judgments are believed to be most likely subjective. Therefore, I thought to reclassify the review into three classes instead of 5. In the new classification, I relabelled the data so that reviews that were classified as1 and 2 are considered bad, 4 and 5 were labeled as good, and 3 were neutral. After training the models, the logistic regression results went up to 87% accuracy. However, I kept the final test set as asked in the original task and used the original 5-class classification model to classify the reviews. I also wanted to try Named Entity Recognizer tags using the Stanford NER tagger [10] and add these to the features but due to time I couldn't.

BERT

I further tried the state-of-the-art pre-trained mode Bert [1]. However, there were some bugs that because of the limited time, I couldn't solve, you can look at the code anyway on the project repository. I used Bert before in another study, if you would like to see my other work using text classification using Bert you check my GitHub repository here [2].

The Code:

The code with a readme file on how to run it can be found in [3].

References:

[1] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[2] https://github.com/Maj27/HealthNewsReview/blob/master/Health%20News%20Review%20using%20BERT%20and%20TF2.ipynb

[3] https://github.com/Maj27/HotelReview

[4] Mitchell, T. M. (1997). Does machine learning really work?. *AI magazine*, *18*(3), 11-11.

[5] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22.

[6] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

[7] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. In *Advances in neural information processing systems* (pp. 6638-6648).

[8] Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[9] Al-Jefri, M., Evans, R., Lee, J., & Ghezzi, P. (2020). Automatic Identification of Information Quality Metrics in Health News Stories. *Frontiers in Public Health*, *8*, 953.

[10] Finkel, J. R., Grenager, T., & Manning, C. D. (2005, June). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)* (pp. 363-370).

[11] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems* (pp. 4765-4774).