

CS506

Hasan Mustafabayli

Lance Galletti

8 November, 2023

Username: Hasan Mustafabayli

Midterm Report

Introduction:

The main objective of this project is to predict customer sentiment based on the given features. I will describe our approach for feature selection, model selection, and model validation.

Finding Key Features:

a. Data exploration:

Utilizing the training and testing datasets, the algorithm displays summary statistics, searches for missing values, and generates visualizations to allow users to explore the data. The most and lowest rated items, top and bottom reviewers, mean helpfulness numerator per score, kindest and harshest reviewers, and the mean score of the top 25 most rated products are all represented visually in bar plot form.

b. Feature engineering:

By computing helpfulness, review and summary length, review year, and sentiment analysis using TextBlob, the algorithm generates new features. Additionally, it employs TfidfVectorizer to generate TF-IDF features for the review text and summary and generates binary features for positive reviews.

c. Feature selection techniques:

After removing unnecessary columns like Id, ProductId, UserId, Text, and Summary, the code divides the training set into training and testing sets in order to execute feature selection. The model's performance is then assessed by fitting an X model to the training set and making predictions on the testing set.

Model Selection:

Comparing the dataset performance of multiple candidate models is how I go about choosing the best model. Here, I've applied the XGBoost model. But I also take into account a range of models, such as Random Forests, Decision Trees, Support Vector Machines, Gradient Boosting Machines (GBMs), and LightGBM. Using cross-validation, we train each model on the dataset and evaluate the results in terms of accuracy, precision, recall, F1-score, and other pertinent metrics.

I choose the model that works best on my dataset based on the performance comparison, making sure to strike a balance between interpretability and model complexity.

Hyperparameter Tuning:

I used `RandomizedSearchCV` to construct a randomized search for hyperparameter tuning. This is a useful method for navigating the hyperparameter space without requiring the laborious calculation that a grid search requires. I created a parameter grid where I could adjust the number of estimators, learning rate, tree depth, and other XGBoost algorithm-specific factors. I minimized overfitting and made sure my hyperparameter tuning procedure was reliable by employing cross-validation with three folds.

Model Validation:

I do the following actions to assess the performance of the finished model and make sure it can be generalized:

- a. I have used a training set and testing set at the same time for accuracy.
- b. Performance metrics: I compute various performance metrics, such as confusion matrix, to assess the model's performance on the test set. In this case, I have used root mean squared error (RMSE).

Lastly, I used `pickle` to store my best XGBoost model so I could quickly reload it for predictions in the future without having to retrain. These tools are essential for preparing new data in the same manner as the training data.

SEARCH

midterm-hasanmustafabayli

base (Python 3.11.4)

starter_code.ipynb / 11

requirements.txt

starter_code-2.ipynb

modified_starter_code.ipynb

starter_code_updated_kr.ipynb

starter_code_updated_kr.ipynb - Downloads

starter_code_updated_kr.ipynb /

updated_starter_code.ipynb

...

+

Code

+

Markdown

+

Run All

+

Restart

+

Clear All Outputs

+

Variables

+

Outline

...

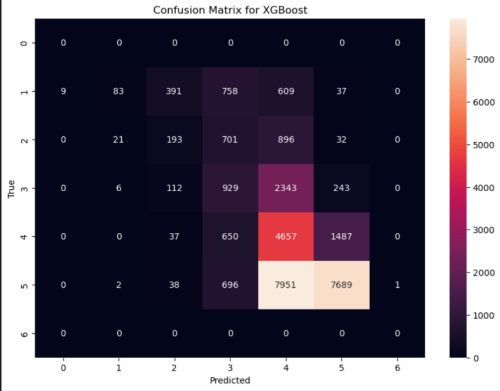
Warning: 'early_stopping_rounds' in 'fit' method is deprecated for better compatibility with scikit-learn, use 'early_stopping_rounds' in const

warnings.warn!

MSE for XGBoost: 0.9633241321487368

Accuracy for XGBoost: 0.4432632233162147

Confusion Matrix for XGBoost



	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	9	83	391	758	609	37	0
2	0	21	193	701	896	32	0
3	0	6	112	929	2343	243	0
4	0	0	37	650	4657	1487	0
5	0	2	38	696	7951	7689	1
6	0	0	0	0	0	0	0

Create the Kaggle submission

```
import pandas as pd
import pickle
import os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
from xgboost import XGBRegressor

# Load the trained XGBoost model
output_dir = "/data"
xgb_model_filename = os.path.join(output_dir, 'xgb_model.pkl')

with open(xgb_model_filename, 'rb') as file:
    best_xgb_model = pickle.load(file)
```

Ln 106, Col 30 Spaces: 4 LF Cell 6 of 11 24 Spd