CS506 Midterm Report

StuID:u25549843

StuName:Yifei Zhou

KaggleName: **YF Zhou8888**

11/06/2023

Goal

The objective of the competition was to predict the star rating (score) tied to user evaluations on Amazon for movies by analyze the provided attributes - 'Id', 'ProductId', 'UserId', 'HelpfulnessNumerator', 'HelpfulnessDenominator', 'Time', 'Summary', 'Text', and 'Score'.
By extract useful information and select a better model ,we can get better result .

Feature Extraction

By observe the 11 plot attached with the file, I understood that:
1.The count of score may be a useful information since score of 5 is far more than other scores
2.The productID may be useful , because for different product ,people seem to show different willingness to rate.
3.The userID may be useful, because for reviewer who is likely to give better score on one product, he or she will also likely to give a better score on another product.
4.There are potential relationship between helpfulness and score
5.There are potential relationship between score and the text written by user. We can analyze the word they use to get a better training process

Additionally, when I take look at the csv file, I notice that some col may be useless, for example time, it is also a potential challenge for this training, it may contain useful information, but it may also decrease the accuracy if it is directly used without any modify.

Based on my observation, I modify the trainset as the code at below,

```
trainingSet['Helpfulness'] = trainingSet['HelpfulnessNumerator'] / trainingSet['HelpfulnessDenominator']
mean_denominator = trainingSet[trainingSet['Helpfulness'] != 0]['Helpfulness'].mean()
trainingSet['Helpfulness'] = trainingSet['Helpfulness'].fillna(mean_denominator)
trainingSet['ReviewLength'] = trainingSet.apply(lambda row : len(row['Text'].split()) if type(row['Text']) == str else 0, axis = 1)
```

I take ProductID as useful information. the model can not take id with char such as ABC, so I

encode them:

```
#'ProductId'
#LabelEncoder
label_encoder = LabelEncoder()
#ProductId to code
trainingSet['ProductId_encoded'] = label_encoder.fit_transform(trainingSet['ProductId'])
```

Analysis based on words used in text:

```
good_words = ['interesting','Interesting', 'classic','Classic', 'best','Best', 'enj
bad_words = ['superficial','Superficial','terrible','Terrible','obnoxious','Obnoxio
negation = ['not','Not','too little', 'Too little',"couldn't", "Couldn't","can't","

trainingSet["negation_count"] = trainingSet['Text'].str.count('|'.join(negation))
trainingSet['negation_count'] = trainingSet['negation_count'].fillna(0)
trainingSet["good_count"] = trainingSet['Text'].str.count('|'.join(good_words))
trainingSet["good_count"] = trainingSet['good_count'].fillna(0)
trainingSet["bad_count"] = trainingSet['Text'].str.count('|'.join(bad_words))
trainingSet["bad_count"] = trainingSet['bad_count'].fillna(0)
```

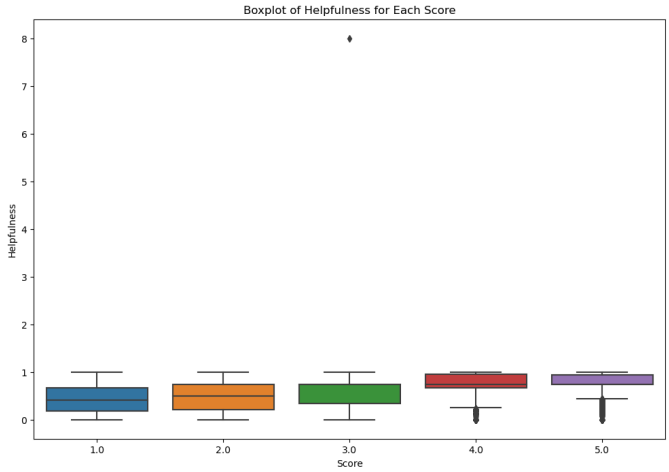
User information:

```
trainingSet['same_user'] = trainingSet.duplicated(subset = ['UserId'])
trainingSet['returning_user'] = trainingSet.apply(lambda row: 1 if row['same_user']== True else 0, axis =1)
trainingSet['same_movie'] = trainingSet.duplicated(subset = ['ProductId'])
trainingSet['movie_repeated']  = trainingSet.apply(lambda row: 1 if row['same_movie'] == True else 0, axis = 1)
```

Review length and sentiment analysis

```
# Review length in characters
trainingSet['ReviewCharLength'] = trainingSet['Text'].apply(lambda x: len(x) if isinstance(x, str) else 0)
# Count of capitalized words
trainingSet['CapitalizedCount'] = trainingSet['Text'].apply(lambda x: sum(map(str.isupper, x.split())) if isinstance(x, str) else 0)
# Sentiment Analysis
trainingSet['Polarity'] = trainingSet['Text'].apply(lambda x: TextBlob(x).sentiment.polarity if isinstance(x, str) else 0)
trainingSet['Subjectivity'] = trainingSet['Text'].apply(lambda x: TextBlob(x).sentiment.subjectivity if isinstance(x, str) else 0)
```

relationship between helpfulness and score



Find Model

Model selection:

#model=RandomForestRegressor()
model=GradientBoostingRegressor()
#model = LinearRegression()
#model=Ridge(alpha=7)

#model=HuberRegressor(alpha=2)

Try different model to find a better one, I use multiple models. At the first there are KNN ,but the performance are not great as I want. When using LinearRegression, it improve a lot. And according to the final performance, Ridge and RandomForestRegressor can show a stable performance, including a lightly improvement. HuberRegression is not better than LinearRegression in this experienment. The best performance I have is GradientBoostingRegression.

For the model, I can only say each algorithm has its strengths and weaknesses, and their performance can vary depending on the nature of the data and the task.

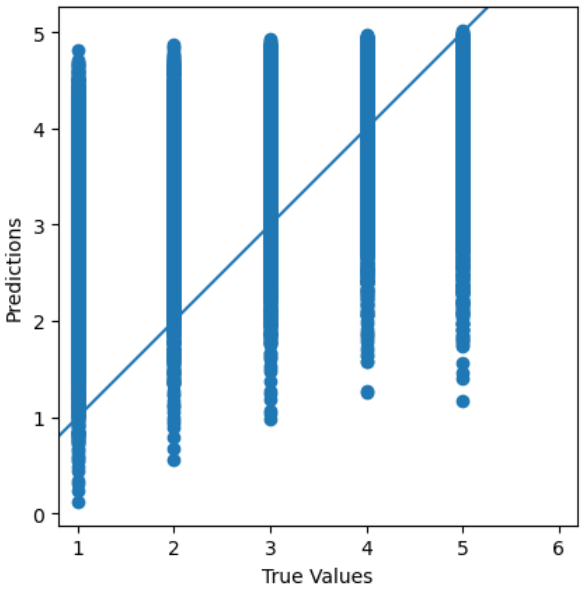
Decision Factors(GBR VS LR):

Data Size and Complexity: For small and simple datasets, a linear model might suffice and will be faster to run. For complex, large datasets with non-linear relationships, gradient boosting usually outperforms linear models.
Performance: If predictive performance is the only concern and the model does not overfit, gradient boosting is likely to perform better.
Training Time: Gradient boosting models can take a lot of time and computational resources to train, especially on large datasets.

Performance Evaluation:

For classification tasks, use metrics like accuracy, F1 score, confusion matrix, etc. For regression, use metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared(here we calculate RMSE).

RMSE on testing set
= 0.9322875053154894



Challenge

when find a feature to train the model， the feature not always help to get a better result.