In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import Ridge
from sklearn.metrics import accuracy_score, mean_squared_error
import matplotlib.pyplot as plt
import pickle
from sklearn.metrics import r2_score, mean_absolute_error
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import pickle
from scipy.sparse import hstack
X_train = pd.read_csv("./data/X_train_stemmed.csv")
X_train["SummaryReview"] = X_train["Summary"] + " " + X_train["Text"]

X_train['SummaryReview'].fillna('', inplace=True)

X_train, X_test, Y_train, Y_test = train_test_split(
        X_train.drop(['Score'], axis=1),
        X_train['Score'],
        test_size=1/4,
        random_state=0
)

X_train_processed = X_train.drop(columns=['Id'])
X_test_processed = X_test.drop(columns=['Id'])
```

In [2]:
```
tfidf_transformer = TfidfVectorizer(ngram_range=(1,4))
X_train_tfidf = tfidf_transformer.fit_transform(X_train_processed['SummaryReview'])
```

In [3]:
```
X_train.columns
```

Out[3]:
```
Index(['Unnamed: 0', 'Id', 'ProductId', 'UserId', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Time', 'Summary', 'Text', 'Helpfulness',
       'ReviewLength', 'SummaryLength', 'product_count', 'sentiment_scores',
       'Real_Time', 'Year', 'Month', 'Day', 'Summary_Stemmed', 'Text_Stemmed',
       'neg_scores', 'pos_scores', 'SummaryReview'],
      dtype='object')
```

In [43]:
```
# X_train['AVG_helpful'] = X_train.groupby('ProductId')['Helpfulness'].transform('mean
# X_test['AVG_helpful'] = X_test.groupby('ProductId')['Helpfulness'].transform('mean')
```

In [4]:
```
X_train_data = hstack([X_train_tfidf,
                       np.array(X_train['Helpfulness']).reshape(-1,1),
                       np.array(X_train['HelpfulnessNumerator']).reshape(-1,1),
                       np.array(X_train['HelpfulnessDenominator']).reshape(-1,1),
                       np.array(X_train['sentiment_scores']).reshape(-1,1),
                        np.array(X_train['pos_scores']).reshape(-1,1),
                        np.array(X_train['neg_scores']).reshape(-1,1),
                     ]
                     )
```

In [7]:
```
ridge = Ridge(alpha=0.09)
ridge.fit(X_train_data, Y_train)
```

Out[7]:

```
▾      Ridge
Ridge(alpha=0.09)
```

In [8]:

```
X_test_tfidf = tfidf_transformer.transform(X_test_processed['SummaryReview'])
Y_test_predictions = ridge.predict(hstack([X_test_tfidf,
                                           np.array(X_test['Helpfulness']).reshape(-1,
                                          ,np.array(X_test['HelpfulnessNumerator']).re
                                           np.array(X_test['HelpfulnessDenominator']).
                                            np.array(X_test['sentiment_scores']).resha
                                            np.array(X_test['pos_scores']).reshape(-1,
                                            np.array(X_test['neg_scores']).reshape(-1,
                                          ]
                                         )).clip(1,5)
print("RMSE on testing set = ", mean_squared_error(Y_test, Y_test_predictions)**0.5)
```

```
RMSE on testing set =  0.7319069645653924
```

Test Set

In [9]:

```
test_set = pd.read_csv("./data/X_test_stemmed.csv")
test_set["SummaryReview"] = test_set["Summary"] + " " + test_set["Text"]
test_tfidf = tfidf_transformer.transform(test_set['SummaryReview'])


prediction = ridge.predict(hstack([test_tfidf,
                                    np.array(test_set['Helpfulness']).reshape(-1,1),
                                    np.array(test_set['HelpfulnessNumerator']).reshape
                                   np.array(test_set['HelpfulnessDenominator']).reshap
                                   np.array(test_set['sentiment_scores']).reshape(-1,1
                                    np.array(test_set['pos_scores']).reshape(-1,1),
                                    np.array(test_set['neg_scores']).reshape(-1,1),
                                             ])).clip(1,5)

test_set['Score'] = prediction
submission = test_set[['Id', 'Score']]
display(submission.head())
submission.to_csv("./submissionn24.csv", index=False)
```

	Id	Score
0	786781	3.491702
1	17153	4.695403
2	1557328	3.084285
3	1242666	4.372545
4	1359242	4.644750

Bagging Regressor - Not as good

In [104…

```
from sklearn.ensemble import BaggingRegressor
```

```
regr = BaggingRegressor(estimator=Ridge(),
                        n_estimators=10, random_state=1).fit(X_train_data, Y_train)
```

In [105…

```
# X_test_tfidf = tfidf_transformer.transform(X_test_processed['SummaryReview'])
Y_test_predictions = regr.predict(hstack([X_test_tfidf,
                                           np.array(X_test['Helpfulness']).reshape(-1,
                                          ,np.array(X_test['HelpfulnessNumerator']).re
                                           np.array(X_test['HelpfulnessDenominator']).
                                            np.array(X_test['sentiment_scores']).resha
                                            np.array(X_test['pos_scores']).reshape(-1,
                        np.array(X_test['neg_scores']).reshape(-1,1),
                                          ]
                                         )).clip(1,5)
print("RMSE on testing set = ", mean_squared_error(Y_test, Y_test_predictions)**0.5)
```

```
RMSE on testing set =  0.7499511441259488
```

In [61]:

```
test_set = pd.read_csv("./data/X_test_stemmed.csv")
test_set["SummaryReview"] = test_set["Summary"] + " " + test_set["Text"]
test_tfidf = tfidf_transformer.transform(test_set['SummaryReview'])


prediction = regr.predict(hstack([test_tfidf,
                                    np.array(test_set['Helpfulness']).reshape(-1,1),
                                    np.array(test_set['HelpfulnessNumerator']).reshape
                                   np.array(test_set['HelpfulnessDenominator']).reshap
                                   np.array(test_set['sentiment_scores']).reshape(-1,1
                                    np.array(test_set['pos_scores']).reshape(-1,1),
                                    np.array(test_set['neg_scores']).reshape(-1,1),
                                             ])).clip(1,5)

test_set['Score'] = prediction
submission = test_set[['Id', 'Score']]
display(submission.head())
submission.to_csv("./submissionn24.csv", index=False)
```

	Id	Score
0	786781	3.378939
1	17153	4.567851
2	1557328	3.109433
3	1242666	4.259523
4	1359242	4.947128