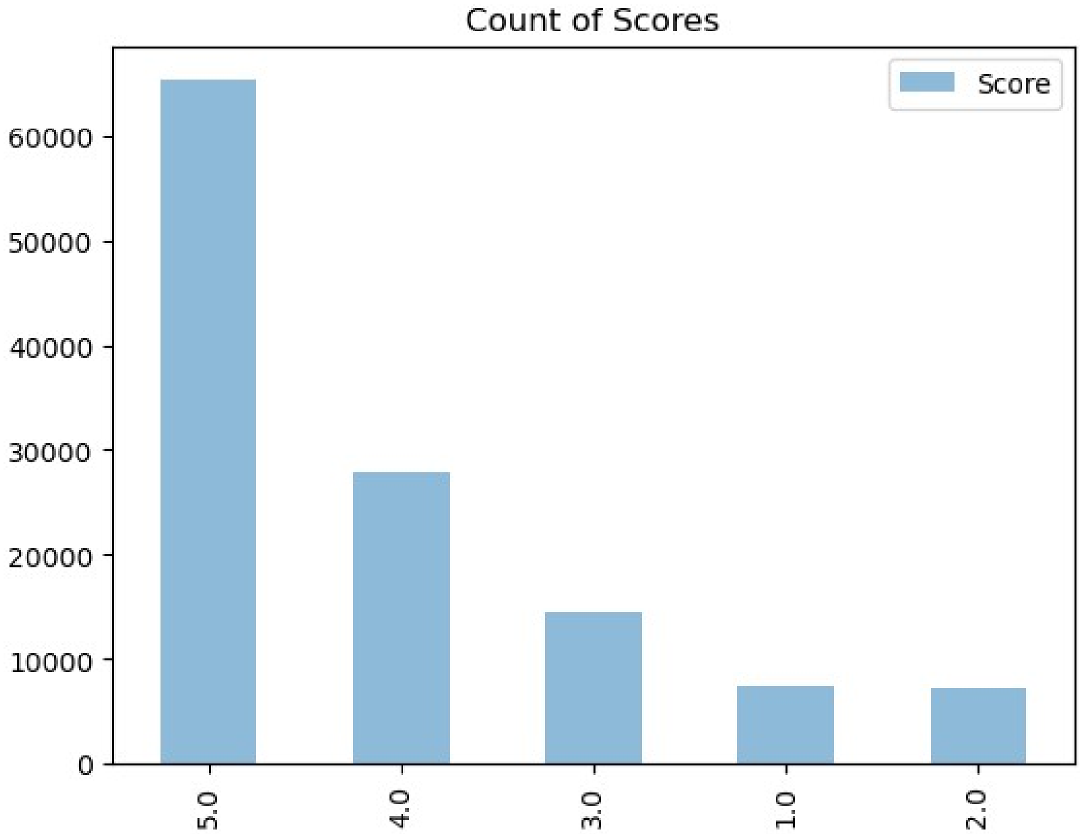
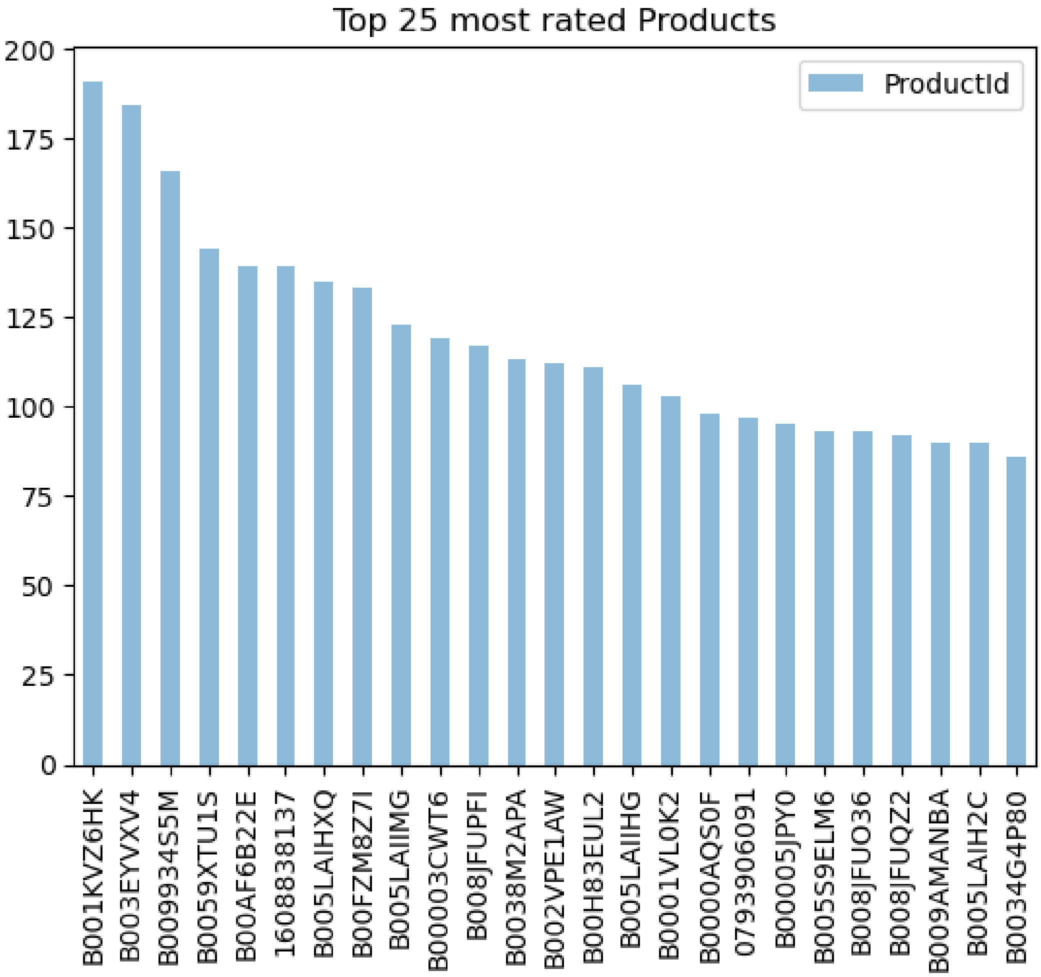
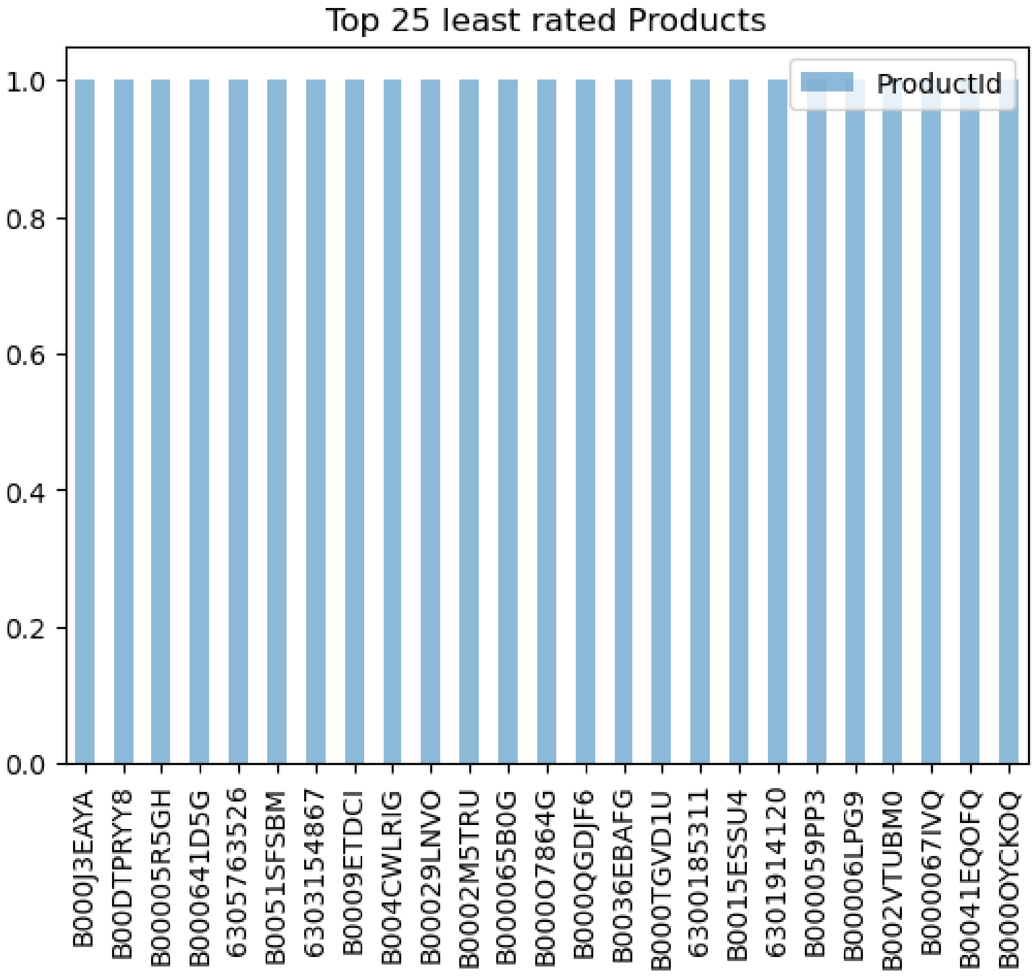
Exploration

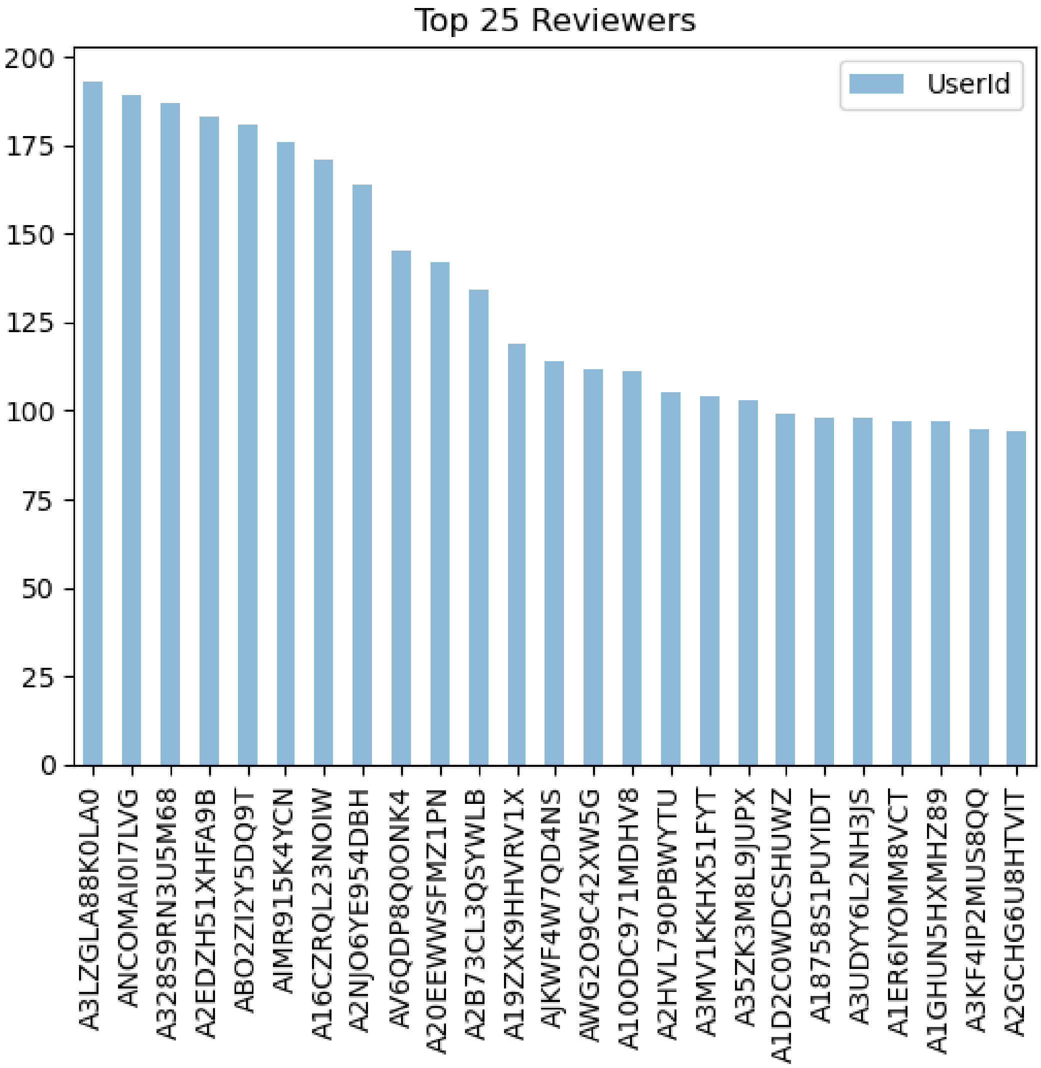
```
import pandas as pd
import matplotlib.pyplot as plt

trainingSet = pd.read_csv("./data/train.csv")
testingSet = pd.read_csv("./data/test.csv")

print("train.csv shape is ", trainingSet.shape)
print("test.csv shape is ", testingSet.shape)

print()

print(trainingSet.head())
print()
print(testingSet.head())

print()

print(trainingSet.describe())

trainingSet['Score'].value_counts().plot(kind='bar', legend=True,
alpha=.5)
plt.title("Count of Scores")
plt.show()

trainingSet['ProductId'].value_counts().nlargest(25).plot(kind='bar',
legend=True, alpha=.5)
plt.title("Top 25 most rated Products")
plt.show()

trainingSet['ProductId'].value_counts().nsmallest(25).plot(kind='bar',
legend=True, alpha=.5)
plt.title("Top 25 least rated Products")
plt.show()

trainingSet['UserId'].value_counts().nlargest(25).plot(kind='bar',
legend=True, alpha=.5)
plt.title("Top 25 Reviewers")
plt.show()

trainingSet['UserId'].value_counts().nsmallest(25).plot(kind='bar',
legend=True, alpha=.5)
plt.title("Lowest 25 Reviewers")
plt.show()

trainingSet[['Score',
'HelpfulnessNumerator']].groupby('Score').mean().plot(kind='bar',
legend=True, alpha=.5)
plt.title("Mean Helpfulness Numerator per Score")
plt.show()

trainingSet[['Score',
'ProductId']].groupby('ProductId').mean().nlargest(25,
'Score').plot(kind='bar', legend=True, alpha=.5)
plt.title("Top 25 best rated Products")
plt.show()

trainingSet[['Score',
'ProductId']].groupby('ProductId').mean().nsmallest(25,
'Score').plot(kind='bar', legend=True, alpha=.5)
plt.title("Top 25 worst rated Products")
plt.show()

trainingSet[['Score', 'UserId']].groupby('UserId').mean().nlargest(25,
'Score').plot(kind='bar', legend=True, alpha=.5)
plt.title("Top 25 kindest Reviewers")
plt.show()

trainingSet[['Score',
'UserId']].groupby('UserId').mean().nsmallest(25,
'Score').plot(kind='bar', legend=True, alpha=.5)
plt.title("Top 25 harshest Reviewers")
plt.show()

trainingSet[trainingSet['ProductId'].isin(trainingSet['ProductId'].val
ue_counts().nlargest(25).index.tolist())][['Score',
'ProductId']].groupby('ProductId').mean().plot(kind='bar',
legend=True, alpha=.5)
plt.title("Mean of top 25 most rated Products")
plt.show()
```

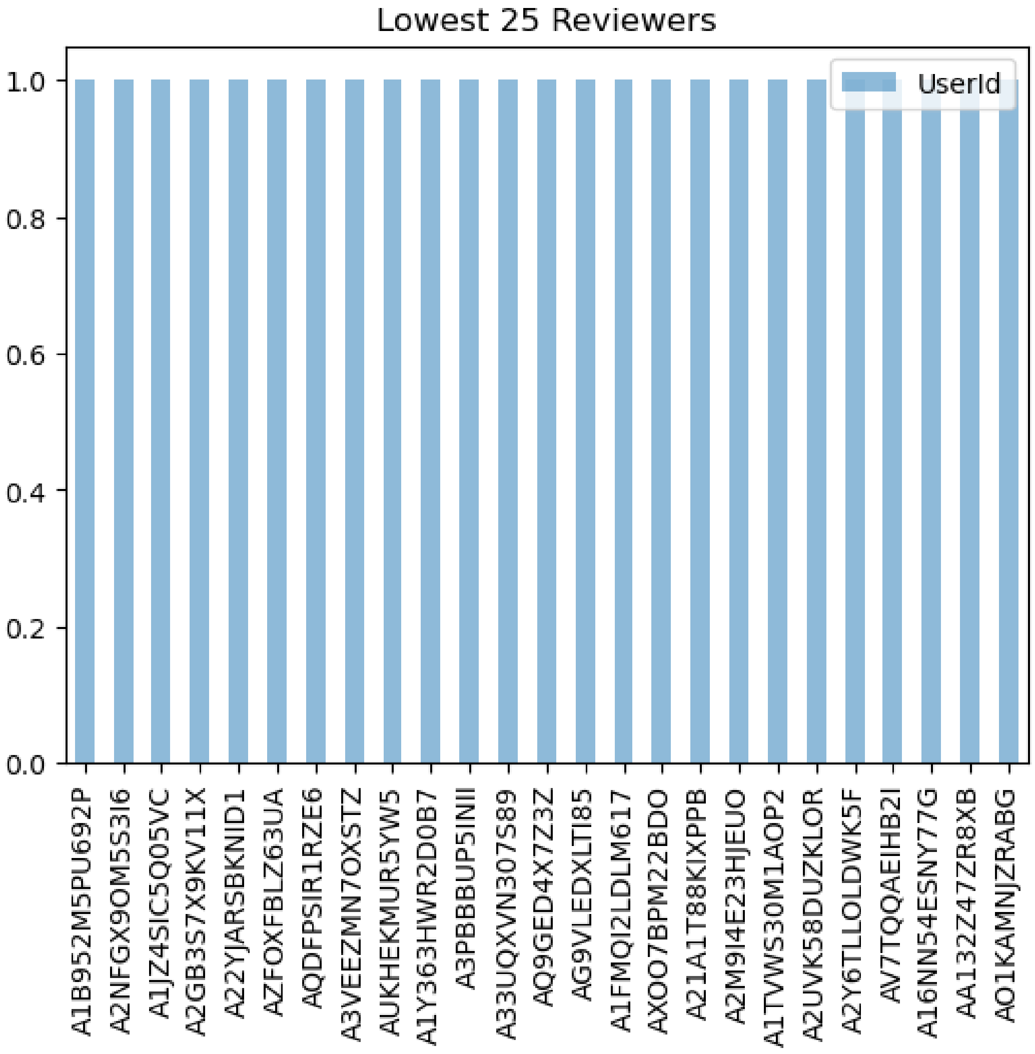
```
train.csv shape is  (139753, 9)
test.csv shape is  (17470, 2)

        Id   ProductId          UserId  HelpfulnessNumerator  \
0   195370  1890228583  A3VLX5Z090RQ0V                     1
1  1632470  B00BEIYSL4   AUDXDMFM49NGY                     0
2     9771  0767809335  A3LFIA97BUU5IE                     3
3   218855  6300215792  A1QZM75342ZQVQ                     1
4   936225  B000B5XOZW   ANM2SCEUL3WL1                     1

   HelpfulnessDenominator        Time  \
0                       2  1030838400
1                       1  1405036800
2                      36   983750400
3                       1  1394841600
4                       1  1163721600

                                             Summary  \
0                         An Unexplained Anime Review
1                                          not great.
2                        Technical problem with this DVD
3                                 Heeeeyyyyy LAAAAADEEE!!!!
4  Herzog the Great Traveler of both natural and ...

                                                Text  Score
0  I was very anxious to see the Uncut version of...    2.0
1                         Movie was okay...not great.    3.0
2  Like the Dinosaur Collector's Edition DVD, thi...    1.0
3  Come on, now.....  this has to be, by far, the...    5.0
4  I've always been a great admirer of Herzog's o...    4.0

        Id  Score
0   786781    NaN
1    17153    NaN
2  1557328    NaN
3  1242666    NaN
4  1359242    NaN

                 Id  HelpfulnessNumerator  HelpfulnessDenominator  \
count  1.397530e+05         139753.000000           139753.000000
mean   8.497881e+05              3.601096                5.313246
std    4.896942e+05             20.101195               22.300962
min    8.000000e+00              0.000000                0.000000
25%    4.258660e+05              0.000000                0.000000
50%    8.510200e+05              1.000000                1.000000
75%    1.273392e+06              3.000000                5.000000
max    1.697519e+06           4646.000000             4682.000000

               Time          Score
count  1.397530e+05  122283.000000
mean   1.262516e+09       4.115552
std    1.287262e+08       1.191661
min    8.948448e+08       1.000000
25%    1.164758e+09       4.000000
50%    1.307318e+09       5.000000
75%    1.373155e+09       5.000000
max    1.406074e+09       5.000000
```

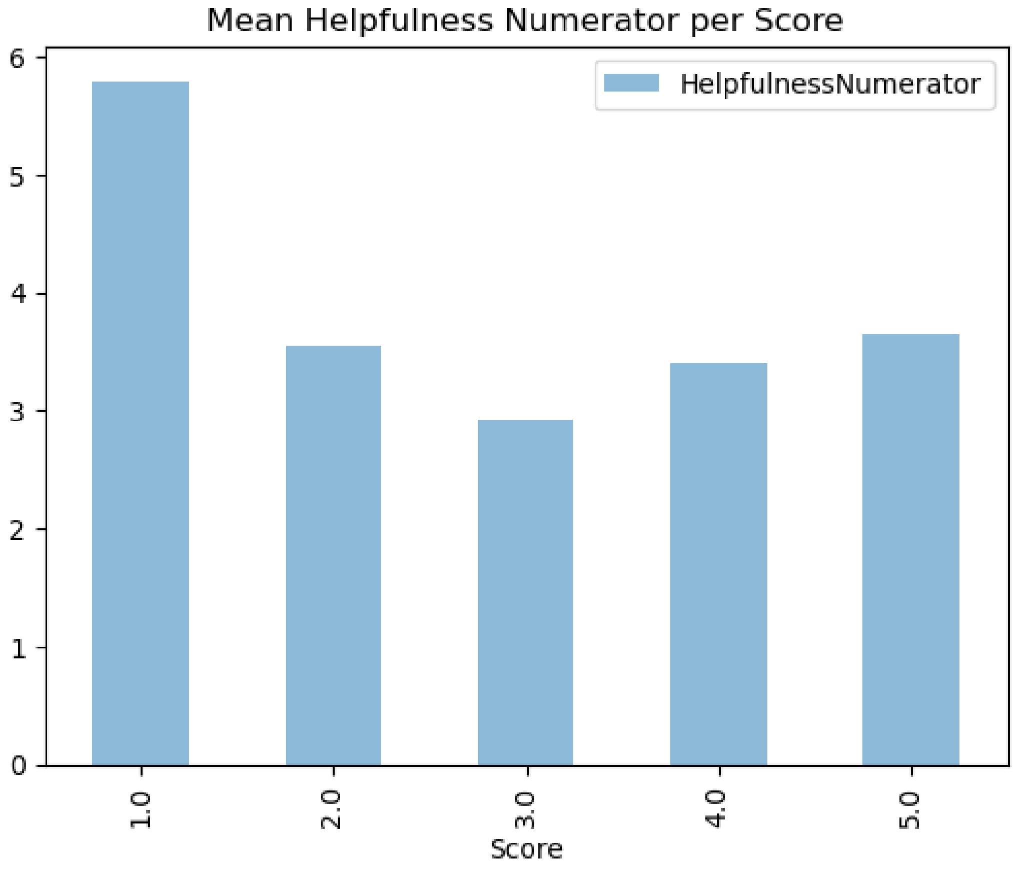

Count of Scores
Score
60000
50000
40000
30000
20000
10000
0
5.0
4.0
3.0
1.0
2.0

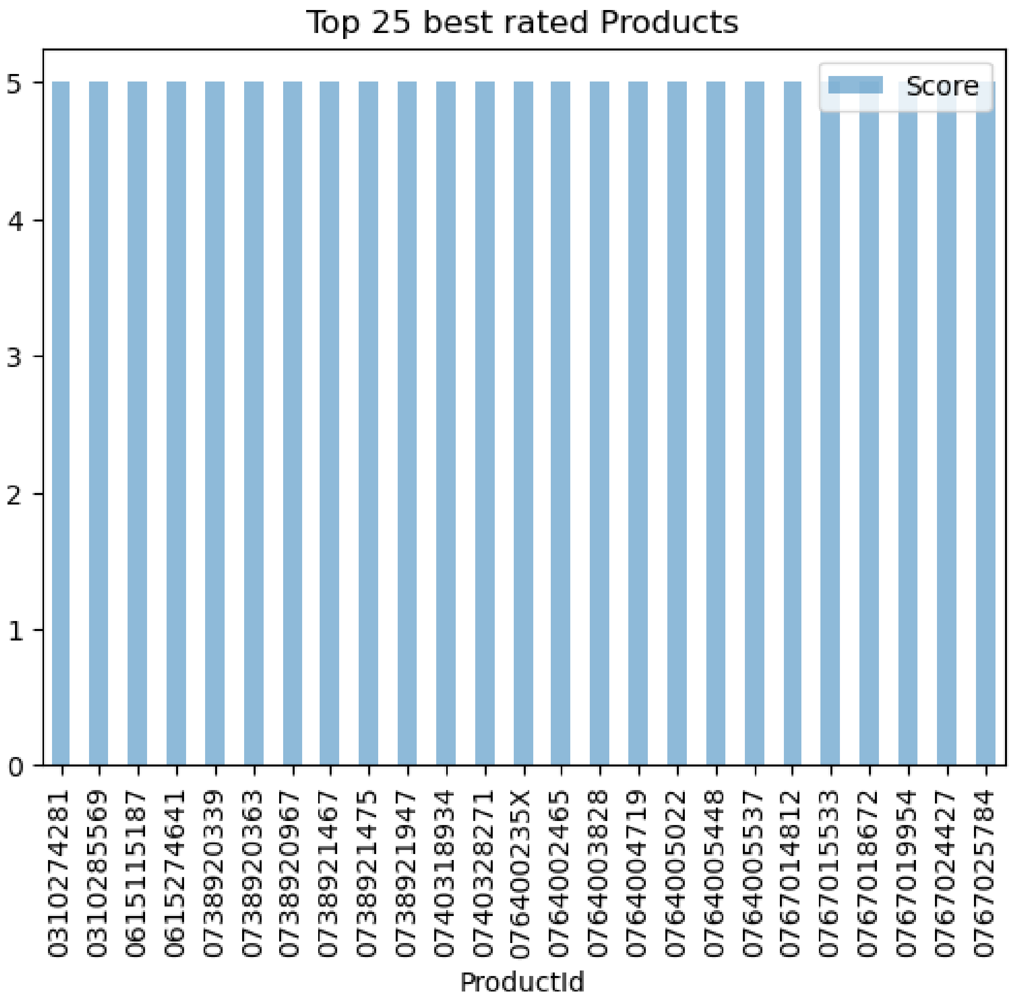

Top 25 most rated Products
ProductId
200
175
150
125
100
75
50
25
0
B001KVZ6HK
B003EYVXV4
B009934S5M
B0059XTU1S
B00AF6B22E
1608838137
B005LAIHXQ
B00FZM8Z7I
B005LAIIMG
B00003CWT6
B008JFUPFI
B0038M2APA
B002VPE1AW
B00H83EUL2
B005LAIIHG
B0001VL0K2
B0000AQS0F
0793906091
B00005JPY0
B005S9ELM6
B008JFUO36
B008JFUQZ2
B009AMANBA
B005LAIH2C
B0034G4P80

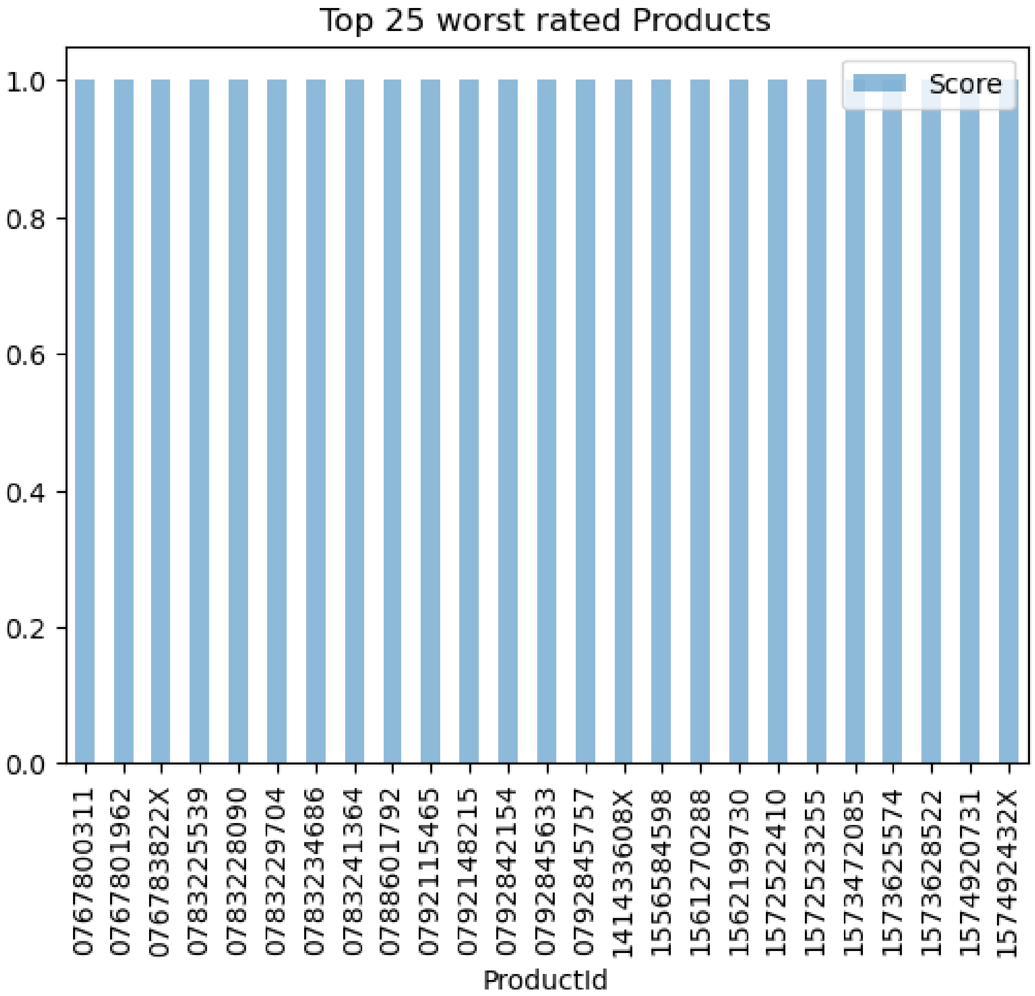

Top 25 least rated Products
ProductId
1.0
0.8
0.6
0.4
0.2
0.0
B000J3EAYA
B00DTPRYY8
B00005R5GH
B000641D5G
6305763526
B0051SFSBM
6303154867
B0009ETDCI
B004CWLRIG
B00029LNVO
B0002M5TRU
B000065B0G
B000O7864G
B000QGDJF6
B0036EBAFG
B000TGVD1U
6300185311
B0015ESSU4
6301914120
B000059PP3
B00006LPG9
B002VTUBM0
B000067IVQ
B0041EQOFQ
B000OYCKOQ

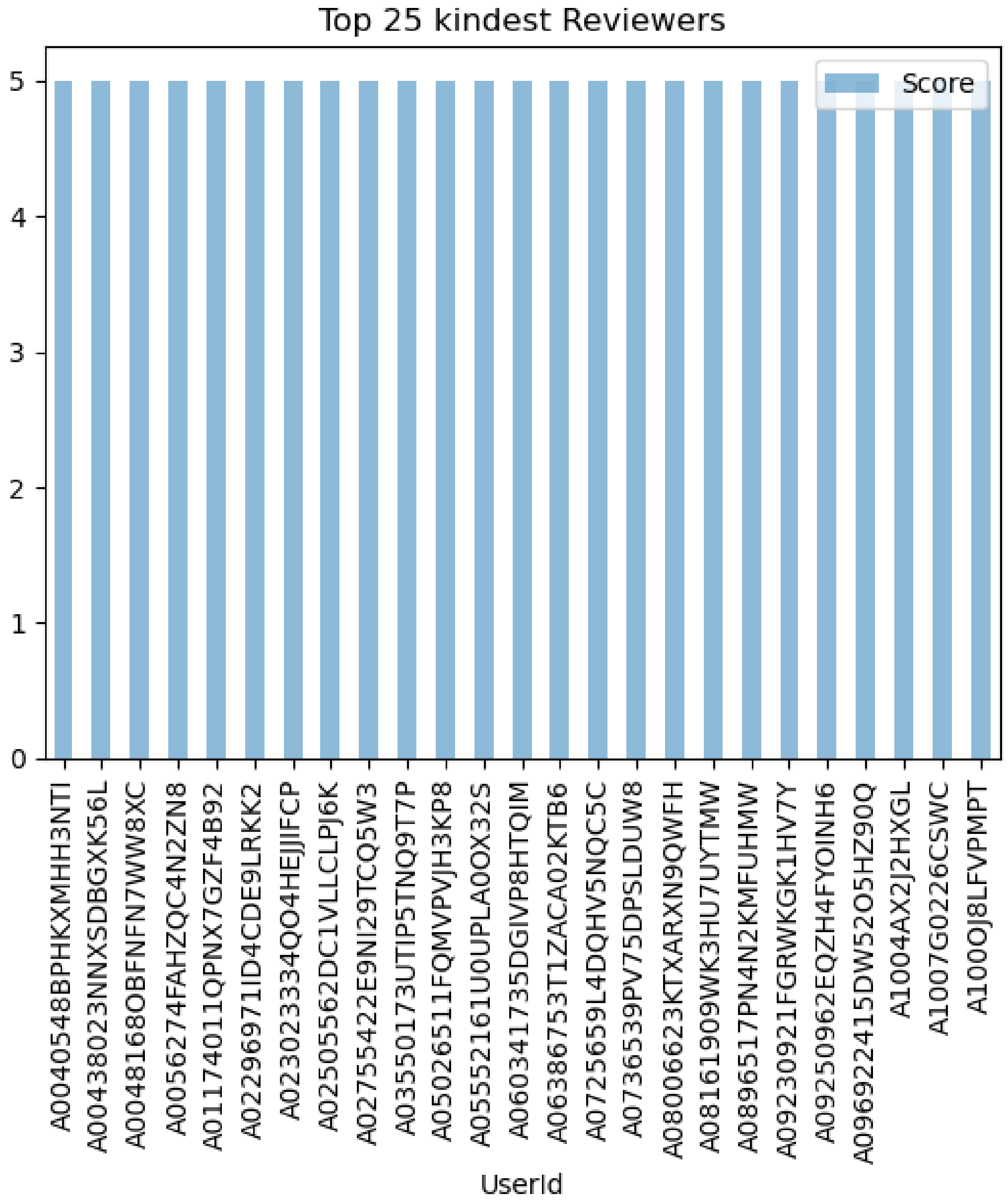

Top 25 Reviewers
UserId
200
175
150
125
100
75
50
25
0
A3LZGLA88K0LA0
ANCOMAI0I7LVG
A328S9RN3U5M68
A2EDZH51XHFA9B
ABO2ZI2Y5DQ9T
AIMR915K4YCN
A16CZRQL23NOIW
A2NJO6YE954DBH
AV6QDP8Q0ONK4
A20EEWWSFMZ1PN
A2B73CL3QSYWLB
A19ZXK9HHVRV1X
AJKWF4W7QD4NS
AWG2O9C42XW5G
A10ODC971MDHV8
A2HVL790PBWYTU
A3MV1KKHX51FYT
A35ZK3M8L9JUPX
A1D2C0WDCSHUWZ
A18758S1PUYIDT
A3UDYY6L2NH3JS
A1ER6IYOMM8VCT
A1GHUN5HXMHZ89
A3KF4IP2MUS8QQ
A2GCHG6U8HTVIT

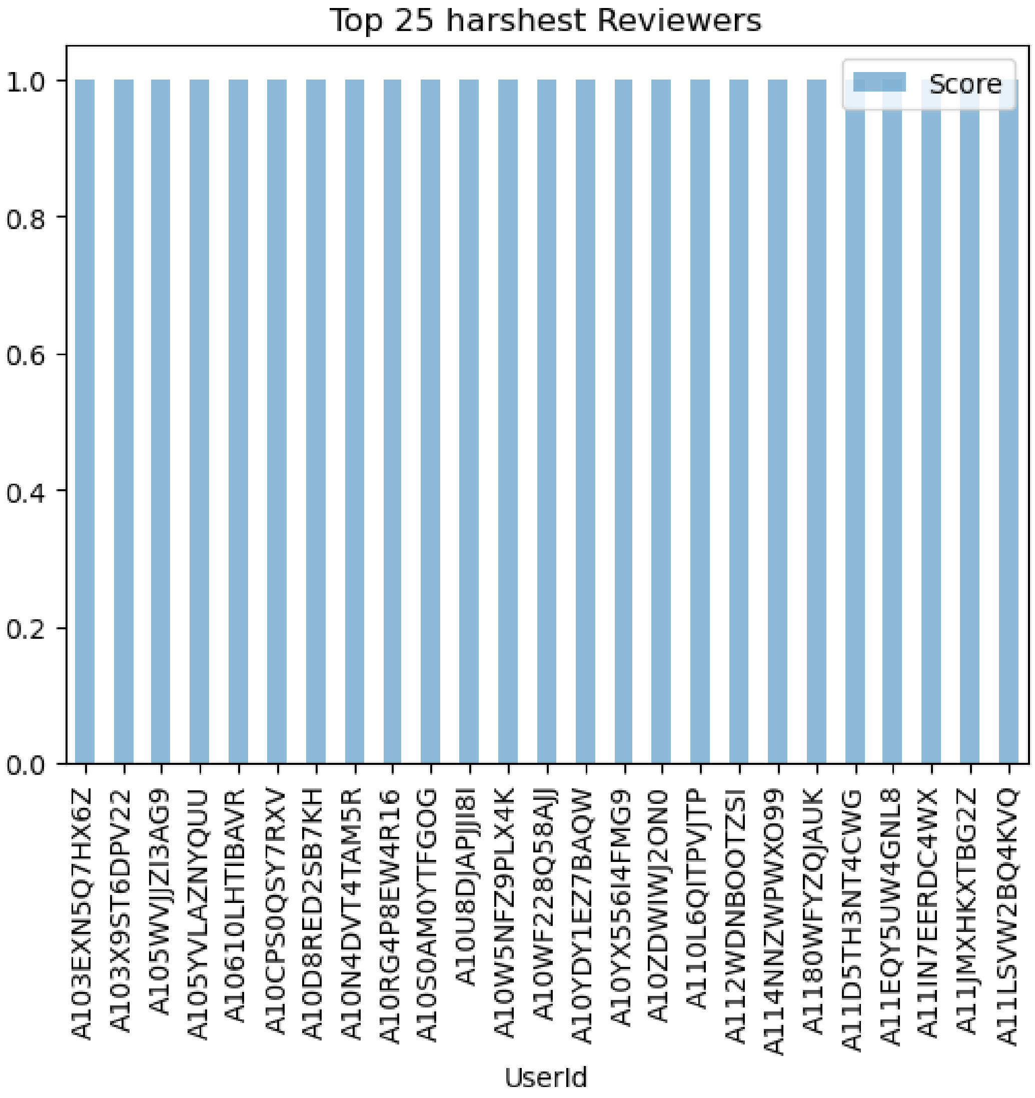

Lowest 25 Reviewers
UserId
1.0
0.8
0.6
0.4
0.2
0.0
A1B952M5PU692P
A2NFGX9OM5S3I6
A1JZ4SIC5Q05VC
A2GB3S7X9KV11X
A22YJARSBKNID1
AZFOXFBLZ63UA
AQDFPSIR1RZE6
A3VEEZMN7OXSTZ
AUKHEKMUR5YW5
A1Y363HWR2D0B7
A3PBBBUP5INII
A33UQXVN307S89
AQ9GED4X7Z3Z
AG9VLEDXLTI85
A1FMQI2LDLM617
AXOO7BPM22BDO
A21A1T88KIXPPB
A2M9I4E23HJEUO
A1TVWS30M1AOP2
A2UVK58DUZKLOR
A2Y6TLLOLDWK5F
AV7TQQAEIHB2I
A16NN54ESNY77G
AA132Z47ZR8XB
AO1KAMNJZRABG

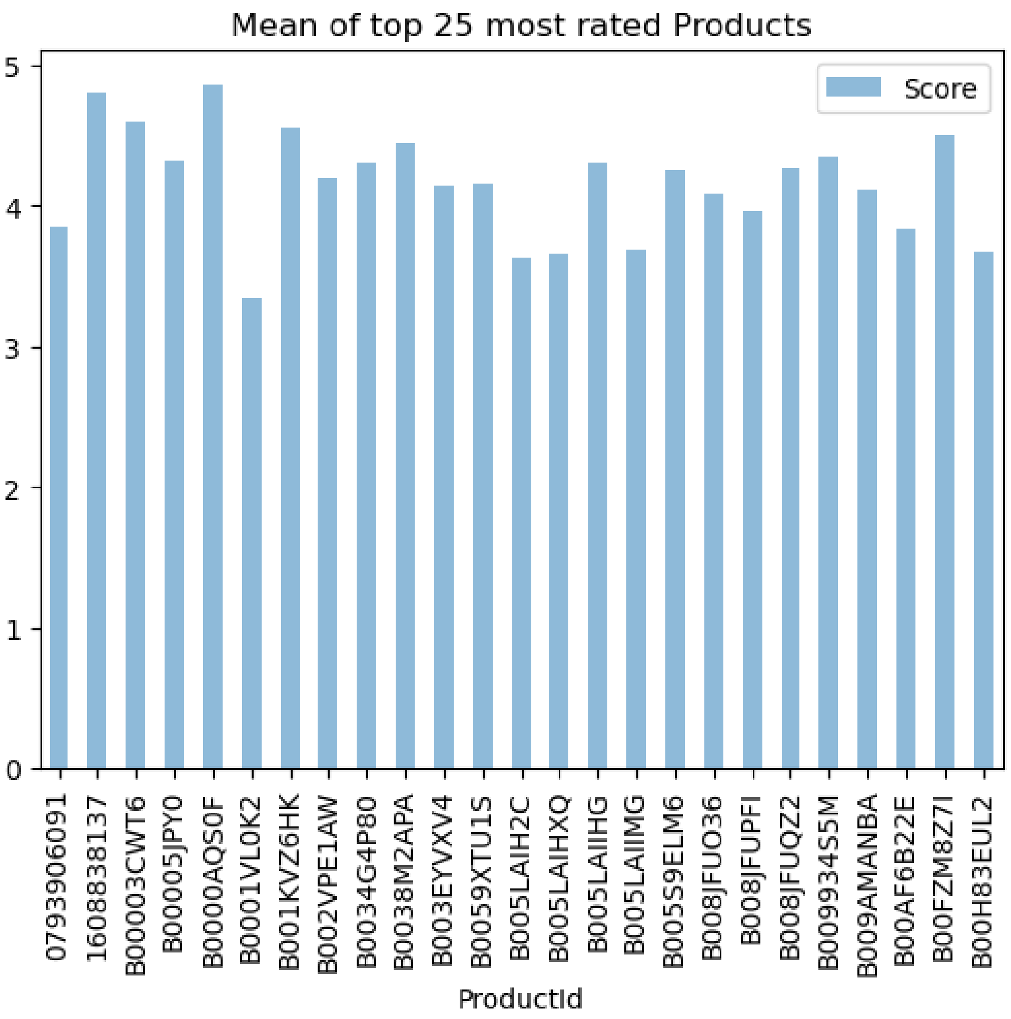

Mean Helpfulness Numerator per Score
HelpfulnessNumerator
6
5
4
3
2
1
0
1.0
2.0
3.0
4.0
5.0
Score


Top 25 best rated Products
Score
0
1
2
3
4
5
0310274281
0310285569
0615115187
0615274641
0738920339
0738920363
0738920967
0738921467
0738921475
0738921947
0740318934
0740328271
076400235X
0764002465
0764003828
0764004719
0764005022
0764005448
0764005537
0767014812
0767015533
0767018672
0767019954
0767024427
0767025784
ProductId


Top 25 worst rated Products
Score
1.0
0.8
0.6
0.4
0.2
0.0
0767800311
0767801962
076783822X
0783225539
0783228090
0783229704
0783234686
0783241364
0788601792
0792115465
0792148215
0792842154
0792845633
0792845757
141433608X
1556584598
1561270288
1562199730
1572522410
1572523255
1573472085
1573625574
1573628522
1574920731
157492432X
ProductId


Top 25 kindest Reviewers
Score
5
4
3
2
1
0
A0040548BPHKXMHH3NTI
A00438023NNXSDBGXK56L
A00481680OBFNFN7WW8XC
A0056274FAHZQC4N2ZN8
A01174011QPNX7GZF4B92
A02296971ID4CDE9LRKK2
A023023334QO4HEJJIFCP
A02505562DC1VLLCLPJ6K
A02755422E9NI29TCQ5W3
A03550173UTIP5TNQ9T7P
A05026511FQMVPVJH3KP8
A05552161U0UPLA0OX32S
A060341735DGIVP8HTQIM
A06386753T1ZACA02KTB6
A07256659L4DQHV5NQC5C
A07365539PV75DPSLDUW8
A08006623KTXARXN9QWFH
A08161909WK3HU7UYTMW
A0896517PN4N2KMFUHMW
A09230921FGRWKGK1HV7Y
A09250962EQZH4FYOINH6
A096922415DW52O5HZ90Q
A1004AX2J2HXGL
A1007G0226CSWC
A100OJ8LFVPMPT
UserId


Top 25 harshest Reviewers
Score
1.0
0.8
0.6
0.4
0.2
0.0
A103EXN5Q7HX6Z
A103X9ST6DPV22
A105WVJJZI3AG9
A105YVLAZNYQUU
A10610LHTIBAVR
A10CPS0QSY7RXV
A10D8RED2SB7KH
A10N4DVT4TAM5R
A10RG4P8EW4R16
A10S0AM0YTFGOG
A10U8DJAPJJI8I
A10W5NFZ9PLX4K
A10WF228Q58AJJ
A10YDY1EZ7BAQW
A10YX556I4FMG9
A10ZDWIWJ2ON0
A110L6QITPVJTP
A112WDNBOOTZSI
A114NNZWPWXO99
A1180WFYZQJAUK
A11D5TH3NT4CWG
A11EQY5UW4GNL8
A11IN7EERDC4WX
A11JMXHKXTBG2Z
A11LSVW2BQ4KVQ
UserId



Feature Extraction

```python
import pandas as pd

def process(df):
    # This is where you can do all your processing

    df['Helpfulness'] = df['HelpfulnessNumerator'] / 
df['HelpfulnessDenominator']
    df['Helpfulness'] = df['Helpfulness'].fillna(0)

    df['ReviewLength'] = df.apply(lambda row : 
len(row['Text'].split()) if type(row['Text']) == str else 0, axis = 1)

    return df

# Load the dataset
trainingSet = pd.read_csv("./data/train.csv")
```

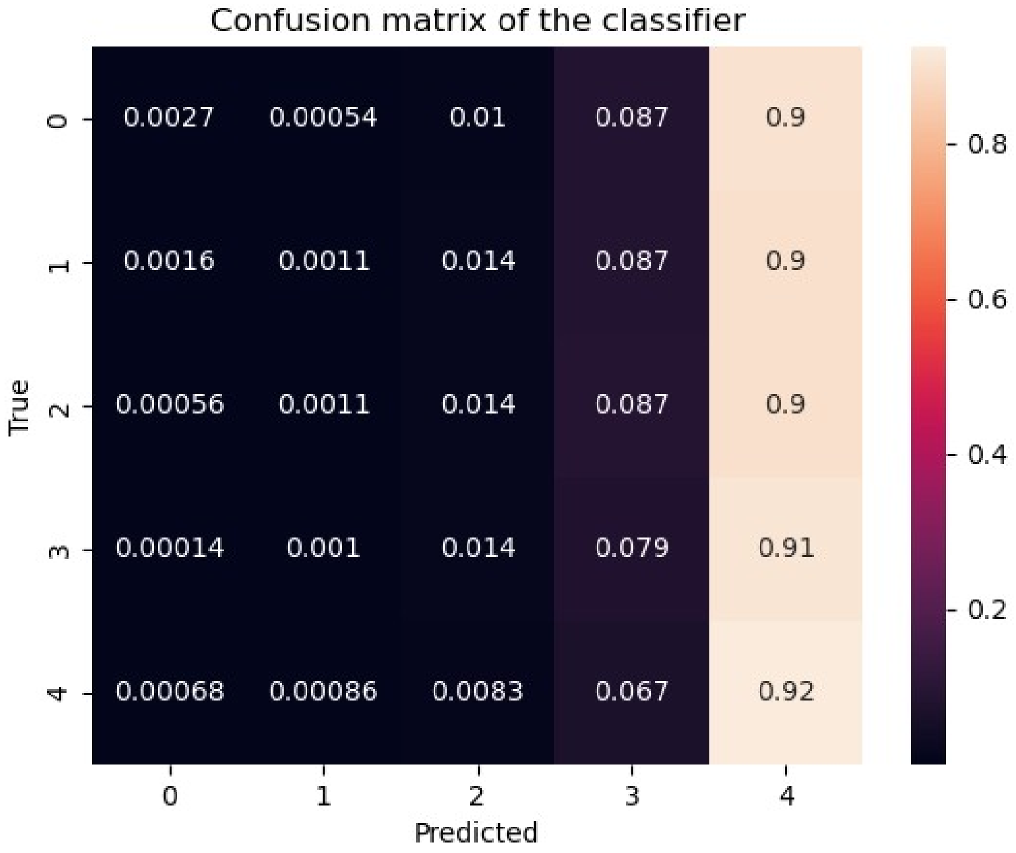
```
# Process the DataFrame
train_processed = process(trainingSet)

# Load test set
submissionSet = pd.read_csv("./data/test.csv")

# Merge on Id so that the test set can have feature columns as well
testX= pd.merge(train_processed, submissionSet, left_on='Id',
right_on='Id')
testX = testX.drop(columns=['Score_x'])
testX = testX.rename(columns={'Score_y': 'Score'})

# The training set is where the score is not null
trainX =  train_processed[train_processed['Score'].notnull()]

# Save the datasets with the new features for easy access later
testX.to_csv("./data/X_test.csv", index=False)
trainX.to_csv("./data/X_train.csv", index=False)
```

Creating your model

```
import pickle
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
mean_squared_error

# Load training set with new features into DataFrame
X_train = pd.read_csv("./data/X_train.csv")

# Split training set into training and testing set
X_train, X_test, Y_train, Y_test = train_test_split(
        X_train.drop(['Score'], axis=1),
        X_train['Score'],
        test_size=1/4.0,
        random_state=3
    )

# This is where you can do more feature selection
X_train_processed = X_train.drop(columns=['Id', 'ProductId', 'UserId',
'Text', 'Summary'])
X_test_processed = X_test.drop(columns=['Id', 'ProductId', 'UserId',
'Text', 'Summary'])

# Learn the model
model = KNeighborsClassifier(n_neighbors=20).fit(X_train_processed,
Y_train)
```

```
# pickle model - saves it so you can load it later
with open('knn_20_model.obj', 'wb') as f:
        pickle.dump(model, f)
# to load pickled model:
# with open('filename', 'rb') as f:
#    model = pickle.load(f)

# Evaluate your model on the testing set
Y_test_predictions = model.predict(X_test_processed)
print("Accuracy on testing set = ", accuracy_score(Y_test,
Y_test_predictions))
print("RMSE on testing set = ", mean_squared_error(Y_test,
Y_test_predictions) ** 0.5)

# Plot a confusion matrix
cm = confusion_matrix(Y_test, Y_test_predictions, normalize='true')
sns.heatmap(cm, annot=True)
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
Accuracy on testing set =  0.5121520395145727
RMSE on testing set =  1.4600992087486073
```



Create the Kaggle submission

```
X_submission = pd.read_csv("./data/X_test.csv")
X_submission_processed = X_submission.drop(columns=['Id', 'ProductId',
'UserId', 'Text', 'Summary', 'Score'])

X_submission['Score'] = model.predict(X_submission_processed)
submission = X_submission[['Id', 'Score']]
submission.to_csv("./data/submission.csv", index=False)
```

Now you can upload the `submission.csv` to kaggle