Exploration

In [46]:

```
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, a
plt.title("Top 25 best rated Products")
plt.show()

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

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

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

trainingSet[trainingSet['ProductId'].isin(trainingSet['ProductId'].value_counts().nlargest(25).index.tolist())][['Scor
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  A3VLX5Z090RQOV                     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
```

Feature Extraction

In [56]:

```
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 =

    """fill NaN text with an empty string to use the new features"""
    df['Text'] = df['Text'].fillna('')

    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

In [60]:

```
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")


# 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'])

"""new features"""
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # You can change max_features based on your dataset size

# Fit and transform the 'Text' column for training data
tfidf_train = tfidf_vectorizer.fit_transform(train_processed['Text'])

# Transform the 'Text' column for test data (Do not fit on test data to avoid data leakage)
tfidf_test = tfidf_vectorizer.transform(train_processed['Text'])

from scipy.sparse import hstack

# Assuming 'other_features' is a DataFrame containing your other features
other_features_train = train_processed.drop(columns=['Id', 'ProductId', 'UserId', 'Text', 'Summary'])
other_features_test = train_processed.drop(columns=['Id', 'ProductId', 'UserId', 'Text', 'Summary'])

# Horizontally stack the sparse tfidf matrix with the other features
X_train_final = hstack([tfidf_train, other_features_train])
X_test_final = hstack([tfidf_test, other_features_test])

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler(with_mean=False)  # Use with_mean=False when working with sparse matrices

X_train_scaled = scaler.fit_transform(X_train_final)
X_test_scaled = scaler.transform(X_test_final)

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

# Learn the model
model = KNeighborsClassifier(n_neighbors=20).fit(X_train, 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_scaled)
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) ** (1/2))

# 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()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_23616\4269847656.py in ?()
     52 
     53 
     54 
     55 # Learn the model
---> 56 model = KNeighborsClassifier(n_neighbors=20).fit(X_train, Y_train)
     57 
     58 # pickle model - saves it so you can load it later
     59 with open('knn_20_model.obj', 'wb') as f:

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py in ?(estimato
r, *args, **kwargs)
   1148                 skip_parameter_validation=(
   1149                     prefer_skip_nested_validation or global_skip_validation
   1150                 )
   1151             ):
-> 1152                 return fit_method(estimator, *args, **kwargs)

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\neighbors\_classifica
tion.py in ?(self, X, y)
    229         -------
    230         self : KNeighborsClassifier
    231             The fitted k-nearest neighbors classifier.
    232         """
--> 233         return self._fit(X, y)

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\neighbors\_base.py in
?(self, X, y)
    453     def _fit(self, X, y=None):
    454         if self._get_tags()["requires_y"]:
    455             if not isinstance(X, (KDTree, BallTree, NeighborsBase)):
--> 456                 X, y = self._validate_data(
    457                     X, y, accept_sparse="csr", multi_output=True, order="C"
    458                 )
    459 

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py in ?(self, X,
y, reset, validate_separately, cast_to_ndarray, **check_params)
    618                 if "estimator" not in check_y_params:
    619                     check_y_params = {**default_check_params, **check_y_params}
    620                 y = check_array(y, input_name="y", **check_y_params)
    621             else:
--> 622                 X, y = check_X_y(X, y, **check_params)
    623             out = X, y
    624 
    625         if not no_val_X and check_params.get("ensure_2d", True):

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py i
n ?(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_n
d, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
   1142         raise ValueError(
   1143             f"{estimator_name} requires y to be passed, but the target y is None"
   1144         )
   1145 
-> 1146     X = check_array(
   1147         X,
   1148         accept_sparse=accept_sparse,
   1149         accept_large_sparse=accept_large_sparse,

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py i
n ?(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_
nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    912                     )
    913                     array = xp.astype(array, dtype, copy=False)
    914                 else:
    915                     array = _asarray_with_order(array, order=order, dtype=dtype, xp=xp)
--> 916             except ComplexWarning as complex_warning:
    917                 raise ValueError(
    918                     "Complex data not supported\n{}\n".format(array)
    919                 ) from complex_warning

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\_array_api.py i
n ?(array, dtype, order, copy, xp)
    376         # Use NumPy API to support order
    377         if copy is True:
    378             array = numpy.array(array, order=order, dtype=dtype)
    379         else:
--> 380             array = numpy.asarray(array, order=order, dtype=dtype)
    381 
    382         # At this point array is a NumPy ndarray. We convert it to an array
    383         # container that is consistent with the input's namespace.

c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\generic.py in ?(s
elf, dtype)
   2082     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
   2083         values = self._values
-> 2084         arr = np.asarray(values, dtype=dtype)
   2085         if (
   2086             astype_is_view(values.dtype, arr.dtype)
   2087            and using_copy_on_write()

ValueError: could not convert string to float: 'B001ILFUDW'
```

Create the Kaggle submission

In [61]:

```
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)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
c:\Users\Yuhan\midterm-davidyh3\starter_code.ipynb Cell 8 line 4
      <a href='vscode-notebook-cell:/c%3A/Users/Yuhan/midterm-davidyh3/starter_code.ipynb#X10sZmlsZQ%3D%3D?line=0'>1
</a> X_submission = pd.read_csv("./data/X_test.csv")
      <a href='vscode-notebook-cell:/c%3A/Users/Yuhan/midterm-davidyh3/starter_code.ipynb#X10sZmlsZQ%3D%3D?line=1'>2
</a> X_submission_processed = X_submission.drop(columns=['Id', 'ProductId', 'UserId', 'Text', 'Summary', 'Score'])
----> <a href='vscode-notebook-cell:/c%3A/Users/Yuhan/midterm-davidyh3/starter_code.ipynb#X10sZmlsZQ%3
D%3D?line=3'>4</a> X_submission['Score'] = model.predict(X_submission_processed)
      <a href='vscode-notebook-cell:/c%3A/Users/Yuhan/midterm-davidyh3/starter_code.ipynb#X10sZmlsZQ%3D%3D?line=4'>5
</a> submission = X_submission[['Id', 'Score']]
      <a href='vscode-notebook-cell:/c%3A/Users/Yuhan/midterm-davidyh3/starter_code.ipynb#X10sZmlsZQ%3D%3D?line=5'>6
</a> submission.to_csv("./data/submission.csv", index=False)

File c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\neighbors\_class
ification.py:266, in KNeighborsClassifier.predict(self, X)
    263         return self.classes_[np.argmax(probabilities, axis=1)]
    264     # In that case, we do not need the distances to perform
    265     # the weighting so we do not compute them.
--> 266     neigh_ind = self.kneighbors(X, return_distance=False)
    267     neigh_dist = None
    268 else:

File c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\neighbors\_base.
py:804, in KNeighborsMixin.kneighbors(self, X, n_neighbors, return_distance)
    802         X = _check_precomputed(X)
    803     else:
--> 804         X = self._validate_data(X, accept_sparse="csr", reset=False, order="C")
    806 n_samples_fit = self.n_samples_fit_
    807 if n_neighbors > n_samples_fit:

File c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:580, in B
aseEstimator._validate_data(self, X, y, reset, validate_separately, cast_to_ndarray, **check_params)
    509 def _validate_data(
    510     self,
    511     X="no_validation",
   (...)
    516     **check_params,
    517 ):
    518     """Validate input data and set or check the `n_features_in_` attribute.
    519 
    520     Parameters
   (...)
    578         validated.
    579     """
--> 580     self._check_feature_names(X, reset=reset)
    582     if y is None and self._get_tags()["requires_y"]:
    583         raise ValueError(
    584             f"This {self.__class__.__name__} estimator "
    585             "requires y to be passed, but the target y is None."
    586         )

File c:\Users\Yuhan\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:507, in B
aseEstimator._check_feature_names(self, X, reset)
    502 if not missing_names and not unexpected_names:
    503     message += (
    504         "Feature names must be in the same order as they were in fit.\n"
    505     )
--> 507 raise ValueError(message)

ValueError: The feature names should match those that were passed during fit.
Feature names seen at fit time, yet now missing:
- Score
```

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