Instrument Reviews, Visualized

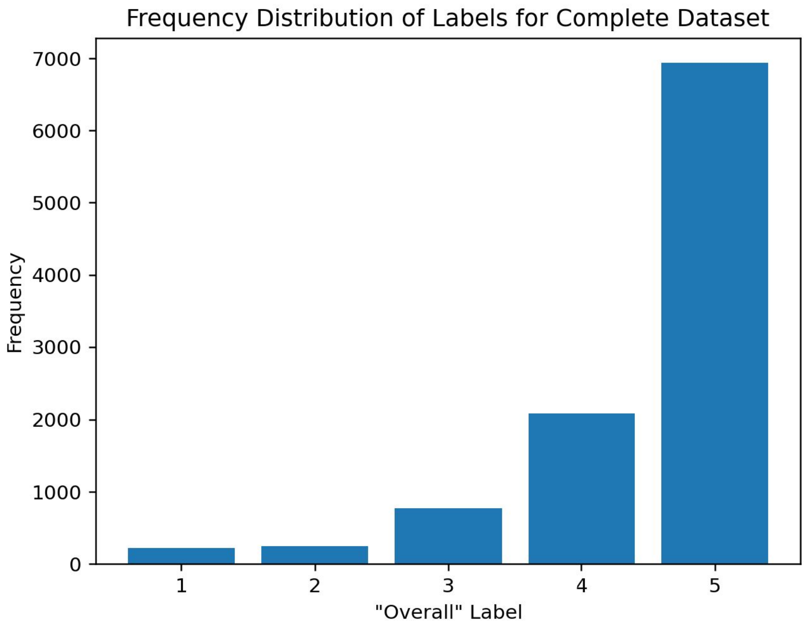
Jake Norbie

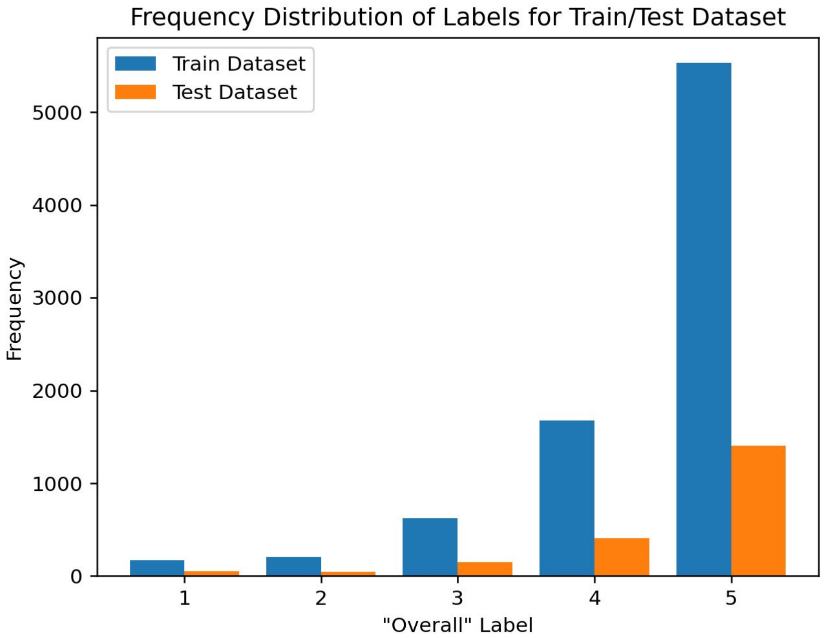
Basic Notes about Dataset

- Amazon Reviews of Instruments
- 10261 documents total
    - 8209/2052 train-test split
- “Review” tag included reviews
    - Highly variable words
    - Required removal of punctuation and splitting of words
- “Overall” tag had values ‘1.0’, ‘2.0’, ‘3.0’, ‘4.0’, and ‘5.0’.
    - Five stars of rating for musical instruments.

Fun Fact: Kaggle says there are 10255 unique reviews, so 6 reviews are identical to existing ones!

Label Distributions





Distribution for labels is not uniform

- Label ‘5.0’ accounted for average 65% of documents.

- Label ‘4.0’ accounted for average 20% of documents.

- Labels ‘1.0’, ‘2.0’, and ‘3.0’ accounted for the rest (~15%).

```
5.0    6938
4.0    2084
3.0     772
2.0     250
1.0     217
Name: overall, dtype: int64
```

Preprocessing Psuedocode

```
df.Reviews = df.Reviews.apply(remove punctuation)

df.Reviews = df.Reviews.apply(split sentence into words)

df.Reviews = df.Reviews.apply(lowercasing)

df.Reviews = df.Reviews.apply(remove stop words in nltk stopwords corpus)

If ignore preprocessing step == False:

     df.Reviews = df.Reviews.apply(PorterStemmer from nltk.stem)

train(df)
```

Training Psuedocode (initial steps)

```
train_data, test_data = split df into 80/20 split

For row in train_data:

     For word in train_data.review:

          If word not in bag_of_words_map:

               bag_of_words_map.append(word)

For label in df.overall.unique:

     Label_prob_dict[label] = count of labels in train data / total
documents
```

Training Psuedocode (building model)

```
Word_probabilities = {}

For target_label in labels:

    Labeled_train_data = train data where label == target_label

    Word_counts = total words in labeled_train_data

    For each word in bag_of_word_map:

        word_probabilities[target_label].append((total word count in
documents with label + 1) / (total word count for label + number of
labels))
```

Testing Psuedocode

```
Test_data.feature = test_data.review.apply(convert review to
bag_of_words vector)

Test_data.prediction = test_data.feature.apply(predict)

Create confusion matrix, metrics, etc.
```

Testing Psuedocode (predict function)

```
Predict_dict = {}

For label:

     For each value in zip(bag_of_word_vector, word_probabilities for label):

          If occurrences != 0:

               Probability *= word probability for label * number of 
occurrences

     Predict_dict[label] = probability * probability of label of occurring

Guess = max(predict_dict)
```

Predicting Psuedocode

```
S = input("Enter your sentence: ")

Feature_vector = preprocessing(S)

Prediction, predict_dict = predict(feature_vector, labels_map,
label_probs, word_probs)

Print probabilities and labels from labels_map, predict_dict
```

Methodologies

- My own code
- Attempted straightforward translation of assignment
- Switched to more ambiguous functions for prediction and testing

The Raw Data

Test results / metrics for label '5.0':

Number of true positives: 1305

Number of false negatives: 71

Number of false positives: 609

Number of true negatives: 67

Sensitivity (recall): 0.9484011627906976

Specificity: 0.09911242603550297

Precision: 0.6818181818181818

Negative predictive value: 0.4855072463768116

Accuracy: 0.6686159844054581

F-Score: 0.7933130699088146

Test results / metrics for label '4.0':

Number of true positives: 34

Number of false negatives: 382

Number of false positives: 102

Number of true negatives: 1534

Sensitivity (recall): 0.08173076923076923

Specificity: 0.9376528117359413

Precision: 0.25

Negative predictive value: 0.8006263048016702

Accuracy: 0.7641325536062378

F-Score: 0.12318840579710146

Test results / metrics for label '3.0':

Number of true positives: 0

Number of false negatives: 174

Number of false positives: 1

Number of true negatives: 1877

Sensitivity (recall): 0.0

Specificity: 0.9994675186368477

Precision: 0.0

Negative predictive value: 0.9151633349585568

Accuracy: 0.9147173489278753

F-Score: 0.0

Test results / metrics for label '2.0':

Number of true positives: 0

Number of false negatives: 47

Number of false positives: 1

Number of true negatives: 2004

Sensitivity (recall): 0.0

Specificity: 0.9995012468827931

Precision: 0.0

Negative predictive value: 0.977084349098001

Accuracy: 0.9766081871345029

F-Score: 0.0

Test results / metrics for label '1.0':

Number of true positives: 0

Number of false negatives: 39

Number of false positives: 0

Number of true negatives: 2013

Sensitivity (recall): 0.0

Specificity: 1.0

Precision: nan

Negative predictive value: 0.9809941520467836

Accuracy: 0.9809941520467836

F-Score: 0.0

The Raw Data (No Stemming)

Test results / metrics for label '5.0':

Number of true positives: 1352

Number of false negatives: 4

Number of false positives: 671

Number of true negatives: 25

Sensitivity (recall): 0.9970501474926253

Specificity: 0.035919540229885055

Precision: 0.6683143845773604

Negative predictive value: 0.8620689655172413

Accuracy: 0.6710526315789473

F-Score: 0.8002367564368156

Test results / metrics for label '4.0':

Number of true positives: 11

Number of false negatives: 438

Number of false positives: 17

Number of true negatives: 1586

Sensitivity (recall): 0.024498886414253896

Specificity: 0.9893948845913911

Precision: 0.39285714285714285

Negative predictive value: 0.7835968379446641

Accuracy: 0.7782651072124757

F-Score: 0.04612159329140461

Test results / metrics for label '3.0':

Number of true positives: 0

Number of false negatives: 154

Number of false positives: 1

Number of true negatives: 1897

Sensitivity (recall): 0.0

Specificity: 0.9994731296101159

Precision: 0.0

Negative predictive value: 0.9249146757679181

Accuracy: 0.9244639376218323

F-Score: 0.0

Test results / metrics for label '2.0':

Number of true positives: 0

Number of false negatives: 46

Number of false positives: 0

Number of true negatives: 2006

Sensitivity (recall): 0.0

Specificity: 1.0

Precision: nan

Negative predictive value: 0.9775828460038987

Accuracy: 0.9775828460038987

F-Score: 0.0

Test results / metrics for label '1.0':

Number of true positives: 0

Number of false negatives: 47

Number of false positives: 0

Number of true negatives: 2005

Sensitivity (recall): 0.0

Specificity: 1.0

Precision: nan

Negative predictive value: 0.9770955165692008

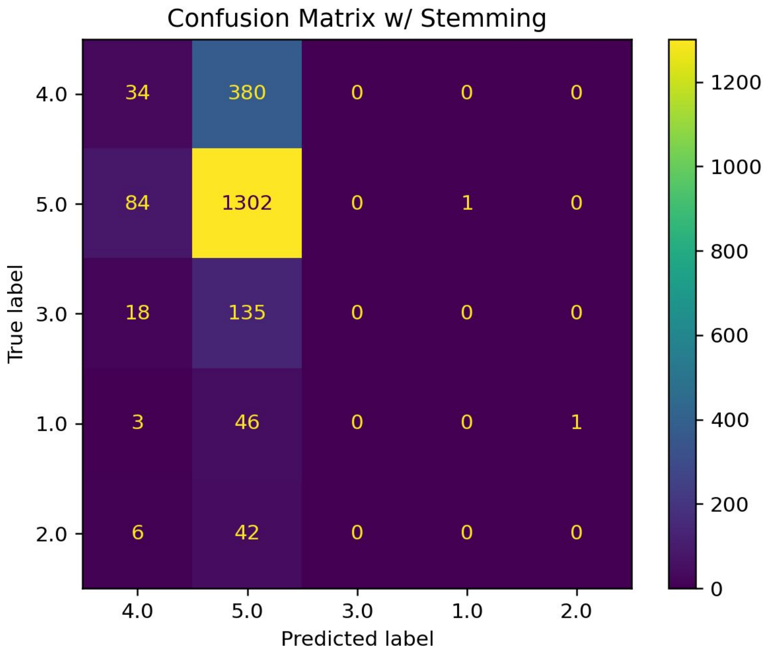
Accuracy: 0.9770955165692008

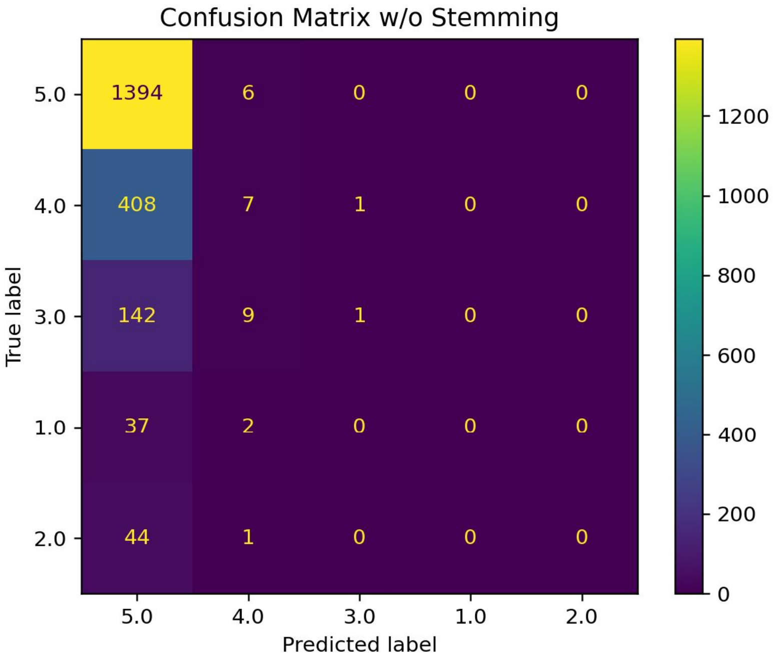
F-Score: 0.0

It’s A Lot, Right?

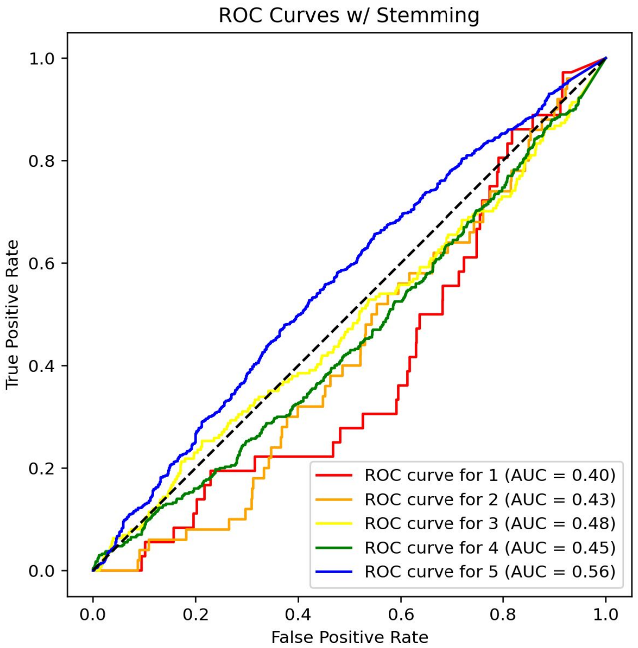
Let’s Break it Down

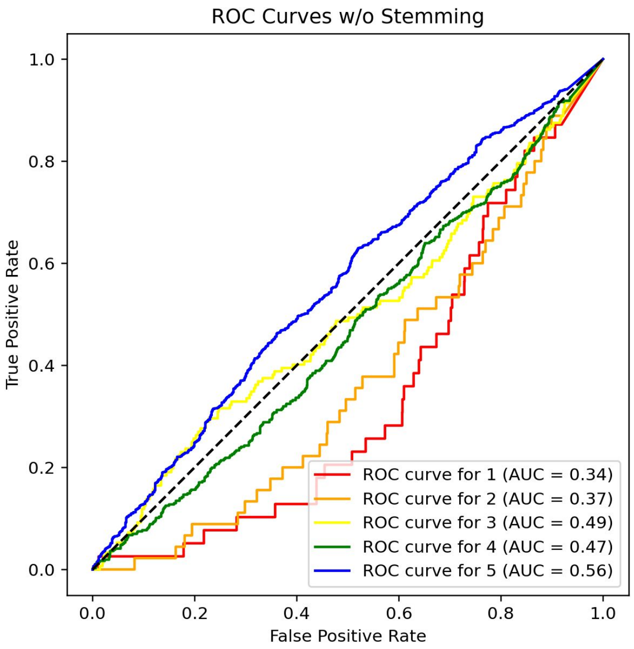
Confusion Matrices





ROC Curves


ROC Curves w/ Stemming
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve for 1 (AUC = 0.40)
ROC curve for 2 (AUC = 0.43)
ROC curve for 3 (AUC = 0.48)
ROC curve for 4 (AUC = 0.45)
ROC curve for 5 (AUC = 0.56)


ROC Curves w/o Stemming
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve for 1 (AUC = 0.34)
ROC curve for 2 (AUC = 0.37)
ROC curve for 3 (AUC = 0.49)
ROC curve for 4 (AUC = 0.47)
ROC curve for 5 (AUC = 0.56)

Distribution for labels is not uniform

- Label ‘5.0’ accounted for average 65% of documents.

- Label ‘4.0’ accounted for average 20% of documents.

- Labels ‘1.0’, ‘2.0’, and ‘3.0’ accounted for the rest (~15%).

```
5.0    6938
4.0    2084
3.0     772
2.0     250
1.0     217
Name: overall, dtype: int64
```

And because of this…

Data weighting is skewed.

If you knew on a multiple choice exam that each question had an 80% chance of having answer ‘a’, you could almost always select ‘a’ on every question and average 80%.

The results show this. Metrics with true negatives are really high, and metrics with positives are really low (almost never predicted).

Try to Guess which Metrics use TN

Test results / metrics for label '5.0':

Number of true positives: 1305

Number of false negatives: 71

Number of false positives: 609

Number of true negatives: 67

Sensitivity (recall): 0.9484011627906976

Specificity: 0.09911242603550297

Precision: 0.6818181818181818

Negative predictive value: 0.4855072463768116

Accuracy: 0.6686159844054581

F-Score: 0.7933130699088146

Test results / metrics for label '4.0':

Number of true positives: 34

Number of false negatives: 382

Number of false positives: 102

Number of true negatives: 1534

Sensitivity (recall): 0.08173076923076923

Specificity: 0.9376528117359413

Precision: 0.25

Negative predictive value: 0.8006263048016702

Accuracy: 0.7641325536062378

F-Score: 0.12318840579710146

Test results / metrics for label '3.0':

Number of true positives: 0

Number of false negatives: 174

Number of false positives: 1

Number of true negatives: 1877

Sensitivity (recall): 0.0

Specificity: 0.9994675186368477

Precision: 0.0

Negative predictive value: 0.9151633349585568

Accuracy: 0.9147173489278753

F-Score: 0.0

Test results / metrics for label '2.0':

Number of true positives: 0

Number of false negatives: 47

Number of false positives: 1

Number of true negatives: 2004

Sensitivity (recall): 0.0

Specificity: 0.9995012468827931

Precision: 0.0

Negative predictive value: 0.977084349098001

Accuracy: 0.9766081871345029

F-Score: 0.0

Test results / metrics for label '1.0':

Number of true positives: 0

Number of false negatives: 39

Number of false positives: 0

Number of true negatives: 2013

Sensitivity (recall): 0.0

Specificity: 1.0

Precision: nan

Negative predictive value: 0.9809941520467836

Accuracy: 0.9809941520467836

F-Score: 0.0

Try to Guess which Metrics use TN

Test results / metrics for label '5.0':

Number of true positives: 1352

Number of false negatives: 4

Number of false positives: 671

Number of true negatives: 25

Sensitivity (recall): 0.9970501474926253

Specificity: 0.035919540229885055

Precision: 0.6683143845773604

Negative predictive value: 0.8620689655172413

Accuracy: 0.6710526315789473

F-Score: 0.8002367564368156

Test results / metrics for label '4.0':

Number of true positives: 11

Number of false negatives: 438

Number of false positives: 17

Number of true negatives: 1586

Sensitivity (recall): 0.024498886414253896

Specificity: 0.9893948845913911

Precision: 0.39285714285714285

Negative predictive value: 0.7835968379446641

Accuracy: 0.7782651072124757

F-Score: 0.04612159329140461

Test results / metrics for label '3.0':

Number of true positives: 0

Number of false negatives: 154

Number of false positives: 1

Number of true negatives: 1897

Sensitivity (recall): 0.0

Specificity: 0.9994731296101159

Precision: 0.0

Negative predictive value: 0.9249146757679181

Accuracy: 0.9244639376218323

F-Score: 0.0

Test results / metrics for label '2.0':

Number of true positives: 0

Number of false negatives: 46

Number of false positives: 0

Number of true negatives: 2006

Sensitivity (recall): 0.0

Specificity: 1.0

Precision: nan

Negative predictive value: 0.9775828460038987

Accuracy: 0.9775828460038987

F-Score: 0.0

Test results / metrics for label '1.0':

Number of true positives: 0

Number of false negatives: 47

Number of false positives: 0

Number of true negatives: 2005

Sensitivity (recall): 0.0

Specificity: 1.0

Precision: nan

Negative predictive value: 0.9770955165692008

Accuracy: 0.9770955165692008

F-Score: 0.0

Results for more popular labels

-Without Stemming is better in 8 of 12 metrics

-Accuracies are roughly the same

-Most metrics are within 0.05 of each other

-Note: different tests revealed different or near opposite results

Conclusions

1) Impartial Distribution of labels without impartial labelling on simple models leads to skewed results, favoring most popular label

2) For this dataset, stemming had very little effect on results