3_0_1_Naive_Bayes_Baseline_Original_Dataset_FINAL

April 13, 2025

[ ]:
```
#@title Install Packages
```

[ ]:
```
!pip install -q transformers
!pip install -q torchinfo
!pip install -q datasets
!pip install -q evaluate
!pip install -q nltk
!pip install -q contractions
!pip install -q hf_xet
!pip install -q sentencepiece
```

[ ]:
```
!sudo apt-get update
! sudo apt-get install tree
```

```
Hit:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64
InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
[3,632 B]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:5 https://r2u.stat.illinois.edu/ubuntu jammy InRelease [6,555 B]
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:8 https://r2u.stat.illinois.edu/ubuntu jammy/main amd64 Packages [2,688 kB]
Hit:9 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages
[1,542 kB]
Hit:11 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy
InRelease
Get:12 https://r2u.stat.illinois.edu/ubuntu jammy/main all Packages [8,824 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3,099
kB]
Hit:14 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Get:15 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages
[2,788 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages
[1,243 kB]
Fetched 20.5 MB in 2s (10.9 MB/s)
```

```
Reading package lists… Done
W: Skipping acquire of configured file 'main/source/Sources' as repository
'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide
it (sources.list entry misspelt?)
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 47.9 kB of archives.
After this operation, 116 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tree amd64 2.0.2-1
[47.9 kB]
Fetched 47.9 kB in 0s (364 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78,
<> line 1.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package tree.
(Reading database … 126315 files and directories currently installed.)
Preparing to unpack …/tree_2.0.2-1_amd64.deb …
Unpacking tree (2.0.2-1) …
Setting up tree (2.0.2-1) …
Processing triggers for man-db (2.10.2-1) …
```

[ ]:
```python
#@title Imports
import nltk
from nltk.tokenize import RegexpTokenizer

import contractions

import evaluate
import transformers
import torch

from torchinfo import summary

from datasets import load_dataset, Dataset, DatasetDict

from transformers import AutoTokenizer, AutoModel,␣
↪AutoModelForSequenceClassification, TrainingArguments, Trainer
```

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import sklearn

import spacy

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report,␣
 ↪precision_recall_fscore_support, accuracy_score

import sentencepiece
```

[ ]:
```
# @title Mount Google Drive
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
dir_root = '/content/drive/MyDrive/266-final/'
# dir_data = '/content/drive/MyDrive/266-final/data/'
# dir_data = '/content/drive/MyDrive/266-final/data/se21-t1-comp-lex-master/'
dir_data = '/content/drive/MyDrive/266-final/data/266-comp-lex-master'
dir_models = '/content/drive/MyDrive/266-final/models/'
dir_results = '/content/drive/MyDrive/266-final/results/'
```

[ ]:
```
log_filename = "experiment_runs.txt"
log_filepath = os.path.join(dir_results, log_filename)
```

[ ]:
```
wandbai_api_key = ""
```

[ ]:
```
!tree /content/drive/MyDrive/266-final/data/266-comp-lex-master/
```

```
/content/drive/MyDrive/266-final/data/266-comp-lex-master/
  fe-test-labels
      test_multi_df.csv
      test_single_df.csv
  fe-train
      train_multi_df.csv
      train_single_df.csv
  fe-trial-val
      trial_val_multi_df.csv
```

```
    trial_val_single_df.csv
  test-labels
    lcp_multi_test.tsv
    lcp_single_test.tsv
  train
    lcp_multi_train.tsv
    lcp_single_train.tsv
  trial
    lcp_multi_trial.tsv
    lcp_single_trial.tsv

6 directories, 12 files
```

[ ]:
```
!ls -R /content/drive/MyDrive/266-final/data/266-comp-lex-master/
```

```
/content/drive/MyDrive/266-final/data/266-comp-lex-master/:
fe-test-labels  fe-train  fe-trial-val  test-labels  train  trial

/content/drive/MyDrive/266-final/data/266-comp-lex-master/fe-test-labels:
test_multi_df.csv  test_single_df.csv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/fe-train:
train_multi_df.csv  train_single_df.csv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/fe-trial-val:
trial_val_multi_df.csv  trial_val_single_df.csv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/test-labels:
lcp_multi_test.tsv  lcp_single_test.tsv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/train:
lcp_multi_train.tsv  lcp_single_train.tsv

/content/drive/MyDrive/266-final/data/266-comp-lex-master/trial:
lcp_multi_trial.tsv  lcp_single_trial.tsv
```

[ ]:
```
!tree /content/drive/MyDrive/266-final/data/266-comp-lex-master/
```

```
/content/drive/MyDrive/266-final/data/266-comp-lex-master/
  fe-test-labels
    test_multi_df.csv
    test_single_df.csv
  fe-train
    train_multi_df.csv
    train_single_df.csv
  fe-trial-val
    trial_val_multi_df.csv
    trial_val_single_df.csv
  test-labels
```

```
      lcp_multi_test.tsv
      lcp_single_test.tsv
   train
      lcp_multi_train.tsv
      lcp_single_train.tsv
   trial
      lcp_multi_trial.tsv
      lcp_single_trial.tsv

6 directories, 12 files
```

[ ]:
```
#@title Import Data
```

[ ]:
```
df_names = [
    "train_single_df",
    "train_multi_df",
    "trial_val_single_df",
    "trial_val_multi_df",
    "test_single_df",
    "test_multi_df"
]

loaded_dataframes = {}

for df_name in df_names:
    if "train" in df_name:
        subdir = "fe-train"
    elif "trial_val" in df_name:
        subdir = "fe-trial-val"
    elif "test" in df_name:
        subdir = "fe-test-labels"
    else:
        subdir = None

    if subdir:
        read_path = os.path.join(dir_data, subdir, f"{df_name}.csv")
        loaded_df = pd.read_csv(read_path)
        loaded_dataframes[df_name] = loaded_df
        print(f"Loaded {df_name} from {read_path}")

# for df_name, df in loaded_dataframes.items():
#     print(f"\n>>> {df_name} shape: {df.shape}")
#     if 'binary_complexity' in df.columns:
#         print(df['binary_complexity'].value_counts())
#         print(df.info())
#         print(df.head())
```

```
for df_name, df in loaded_dataframes.items():
    globals()[df_name] = df
    print(f"{df_name} loaded into global namespace.")
```

```
Loaded train_single_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-train/train_single_df.csv
Loaded train_multi_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-train/train_multi_df.csv
Loaded trial_val_single_df from /content/drive/MyDrive/266-final/data/266-comp-
lex-master/fe-trial-val/trial_val_single_df.csv
Loaded trial_val_multi_df from /content/drive/MyDrive/266-final/data/266-comp-
lex-master/fe-trial-val/trial_val_multi_df.csv
Loaded test_single_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-test-labels/test_single_df.csv
Loaded test_multi_df from /content/drive/MyDrive/266-final/data/266-comp-lex-
master/fe-test-labels/test_multi_df.csv
train_single_df loaded into global namespace.
train_multi_df loaded into global namespace.
trial_val_single_df loaded into global namespace.
trial_val_multi_df loaded into global namespace.
test_single_df loaded into global namespace.
test_multi_df loaded into global namespace.
```

- Functional tests pass, we can proceed with Baseline Modeling

```
[ ]: #@title Experiment 1: Baseline Modeling
```

0.0.1 Reminders:

- Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall

$$\text{Recall} = \frac{TP}{TP + FN}$$

- Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- F1 Score

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Cosine Similarity

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|}$$

- Jaccard Similarity

$$\text{Jaccard Similarity} = \frac{|A \cap B|}{|A \cup B|}$$

- Overlap Similarity (Overlap Coefficient)

$$\text{Overlap Similarity} = \frac{|A \cap B|}{\min(|A|, |B|)}$$

- Dice Coefficient

$$\text{Dice Coefficient} = \frac{2\,|A \cap B|}{|A| + |B|}$$

0.1 Naive Bayes

0.1.1 X = Sentence: contractions and no contractions

- sentence no contractions

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['sentence_no_contractions'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['sentence_no_contractions'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.58      0.74      0.65       229
           1       0.55      0.38      0.44       192

    accuracy                           0.57       421
   macro avg       0.57      0.56      0.55       421
weighted avg       0.57      0.57      0.56       421
```

- sentence with contractions

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence'
X_train = vectorizer.fit_transform(train_df['sentence'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['sentence'])
y_val = val_df['binary_complexity']
```

```
clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.58      0.74      0.65       229
           1       0.55      0.38      0.44       192

    accuracy                           0.57       421
   macro avg       0.57      0.56      0.55       421
weighted avg       0.57      0.57      0.56       421
```

- sentence no contractions

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['sentence_no_contractions'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['sentence_no_contractions'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.52      0.67      0.58        48
           1       0.57      0.41      0.48        51

    accuracy                           0.54        99
   macro avg       0.54      0.54      0.53        99
weighted avg       0.54      0.54      0.53        99
```

- sentence with contractions

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence'
```

```
X_train = vectorizer.fit_transform(train_df['sentence'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['sentence'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.52      0.67      0.58        48
           1       0.57      0.41      0.48        51

    accuracy                           0.54        99
   macro avg       0.54      0.54      0.53        99
weighted avg       0.54      0.54      0.53        99
```

- **Score is higher than expected for a Naive Bayes model**
- **There is no difference in performance when using the input sequence of the sentence with and without contractions**

0.1.2 X = pos_sequence: Part-of-Speech Tags

- POS Tags: Extracts the part-of-speech (POS) tags for each token (e.g., "DET", "NOUN", "VERB").

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(train_df['pos_sequence'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['pos_sequence'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.60      0.67      0.63       229
           1       0.54      0.46      0.50       192
```

```
    accuracy                           0.57       421
   macro avg       0.57      0.57      0.56       421
weighted avg       0.57      0.57      0.57       421
```

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(train_df['pos_sequence'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['pos_sequence'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.58      0.54      0.56        48
           1       0.59      0.63      0.61        51

    accuracy                           0.59        99
   macro avg       0.59      0.58      0.58        99
weighted avg       0.59      0.59      0.59        99
```

- Part of Speech tags outperform raw input sequence

0.1.3 X = dep_sequence: Dependency Tags

- Dependency Tags: Extracts the syntactic dependency labels for each token (e.g., "det", "nsubj", "ROOT").

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(train_df['dep_sequence'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['dep_sequence'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
```

```
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.61      0.60      0.60       229
           1       0.53      0.54      0.54       192

    accuracy                           0.57       421
   macro avg       0.57      0.57      0.57       421
weighted avg       0.57      0.57      0.57       421
```

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(train_df['dep_sequence'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['dep_sequence'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.51      0.46      0.48        48
           1       0.54      0.59      0.56        51

    accuracy                           0.53        99
   macro avg       0.52      0.52      0.52        99
weighted avg       0.52      0.53      0.52        99
```

0.1.4 X = morph_sequence: Morphological Features

- For each token, the morphological attributes have been retrieved for each token

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(train_df['morph_sequence'])
```

```
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['morph_sequence'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.62      0.59      0.60       229
           1       0.53      0.57      0.55       192

    accuracy                           0.58       421
   macro avg       0.58      0.58      0.58       421
weighted avg       0.58      0.58      0.58       421
```

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(train_df['morph_sequence'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['morph_sequence'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.62      0.52      0.57        48
           1       0.61      0.71      0.65        51

    accuracy                           0.62        99
   macro avg       0.62      0.61      0.61        99
weighted avg       0.62      0.62      0.61        99
```

0.1.5 Baseline Experiment Results

Evaluation

- **Raw Sentence Input:** Both with and without contractions, the single-dataset experiment shows a macro F1-score of 0.57, while the multi-dataset experiment yields a lower F1-score (0.54). This suggests that for raw text, model performance degrades on the multi-label version. **While there is no contextual difference between in the contexts between the single and multi versions, the binary_complexity is different, as the complexity scores derived from the 'complex unigram and bigram tokens' in both the single and multi splits of the datasets achieved different scores, and thus different medians (from which we derived our binarized value).**
- **POS Tags:** Using part-of-speech tag sequences produces results similar to raw text on the single dataset (F1 = 0.57) and even slightly better performance on the multi dataset (F1 = 0.59).
- **Dependency Tags:** Dependency label sequences perform on par with the other features in the single-dataset setting (F1 = 0.57) but drop to an F1-score of 0.52 on the multi dataset, indicating less robustness for this representation in that setting.
- **Morphological Features:** On the single dataset, morphological features give a modest improvement (F1 = 0.58) over raw text. Notably, on the multi dataset, they yield the highest performance (F1 = 0.62), suggesting that despite there being no contextual difference between the two, Naive Bayes' capacity to split the complexity of the input sequence is more aligned with the median threshold of the multi-version split of the data. However, it should be noted that the multi-split for trial_val is literally only 99 records, so I expect that these performance metrics will drop substantially on the test set
- **Hyperparameter Tuning:** Naive Bayes was used in a fairly vanilla manner, not reflected in this notebook were some experiments done with varying alphas (i.e. Laplace Smoothing Values)—these led to effectively no difference in average F1 Score results.

Overall, these results indicate that while raw text and simple POS tags are competitive, the morphological feature representation provides an edge—especially in the multi dataset scenario. **This indicates keeping these additional features on-hand for transformers-based ablations may be a good call.**

0.2 Update: Concatenated and Interleaved Features

0.2.1 Single

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_pos_seq'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_pos_seq'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
```

```python
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.59      0.75      0.66       229
           1       0.56      0.38      0.45       192

    accuracy                           0.58       421
   macro avg       0.57      0.56      0.55       421
weighted avg       0.57      0.58      0.56       421
```

[ ]:
```python
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_pos_alt'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_pos_alt'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.59      0.75      0.66       229
           1       0.55      0.37      0.44       192

    accuracy                           0.58       421
   macro avg       0.57      0.56      0.55       421
weighted avg       0.57      0.58      0.56       421
```

[ ]:
```python
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_morph_seq'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_morph_seq'])
y_val = val_df['binary_complexity']
```

```
clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.59      0.78      0.67       229
           1       0.57      0.36      0.44       192

    accuracy                           0.59       421
   macro avg       0.58      0.57      0.56       421
weighted avg       0.58      0.59      0.57       421
```

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_morph_alt'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_morph_alt'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.59      0.78      0.67       229
           1       0.57      0.35      0.44       192

    accuracy                           0.58       421
   macro avg       0.58      0.57      0.55       421
weighted avg       0.58      0.58      0.56       421
```

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_dep_seq'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_dep_seq'])
```

```
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.59      0.75      0.66       229
           1       0.55      0.37      0.44       192

    accuracy                           0.57       421
   macro avg       0.57      0.56      0.55       421
weighted avg       0.57      0.57      0.56       421
```

[ ]:
```
train_df = train_single_df
val_df = trial_val_single_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_dep_alt'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_dep_alt'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.58      0.75      0.66       229
           1       0.55      0.36      0.44       192

    accuracy                           0.57       421
   macro avg       0.57      0.56      0.55       421
weighted avg       0.57      0.57      0.56       421
```

[ ]:

[ ]:

0.2.2 Multi

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_pos_seq'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_pos_seq'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.54      0.69      0.61        48
           1       0.61      0.45      0.52        51

    accuracy                           0.57        99
   macro avg       0.57      0.57      0.56        99
weighted avg       0.57      0.57      0.56        99
```

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_pos_alt'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_pos_alt'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.54      0.69      0.61        48
           1       0.61      0.45      0.52        51

    accuracy                           0.57        99
   macro avg       0.57      0.57      0.56        99
```

```
weighted avg       0.57      0.57      0.56        99
```

[ ]:
```python
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_morph_seq'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_morph_seq'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.58      0.71      0.64        48
           1       0.65      0.51      0.57        51

    accuracy                           0.61        99
   macro avg       0.61      0.61      0.60        99
weighted avg       0.61      0.61      0.60        99
```

[ ]:
```python
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_morph_alt'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_morph_alt'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.57      0.71      0.63        48
           1       0.64      0.49      0.56        51

    accuracy                           0.60        99
```

```
   macro avg       0.60      0.60      0.59        99
weighted avg       0.60      0.60      0.59        99
```

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_dep_seq'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_dep_seq'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.54      0.69      0.61        48
           1       0.61      0.45      0.52        51

    accuracy                           0.57        99
   macro avg       0.57      0.57      0.56        99
weighted avg       0.57      0.57      0.56        99
```

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_dep_alt'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_dep_alt'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.55      0.69      0.61        48
           1       0.62      0.47      0.53        51
```

```
    accuracy                           0.58        99
   macro avg       0.58      0.58      0.57        99
weighted avg       0.58      0.58      0.57        99
```

[ ]:
```
train_df = train_multi_df
val_df = trial_val_multi_df

vectorizer = TfidfVectorizer()  # just on 'sentence_no_contractions'
X_train = vectorizer.fit_transform(train_df['snc_morph_complexity_value'])
y_train = train_df['binary_complexity']

X_val = vectorizer.transform(val_df['snc_morph_complexity_value'])
y_val = val_df['binary_complexity']

clf = MultinomialNB()
clf.fit(X_train, y_train)
preds = clf.predict(X_val)
print(classification_report(y_val, preds))
```

```
              precision    recall  f1-score   support

           0       0.52      0.67      0.58        48
           1       0.57      0.41      0.48        51

    accuracy                           0.54        99
   macro avg       0.54      0.54      0.53        99
weighted avg       0.54      0.54      0.53        99
```