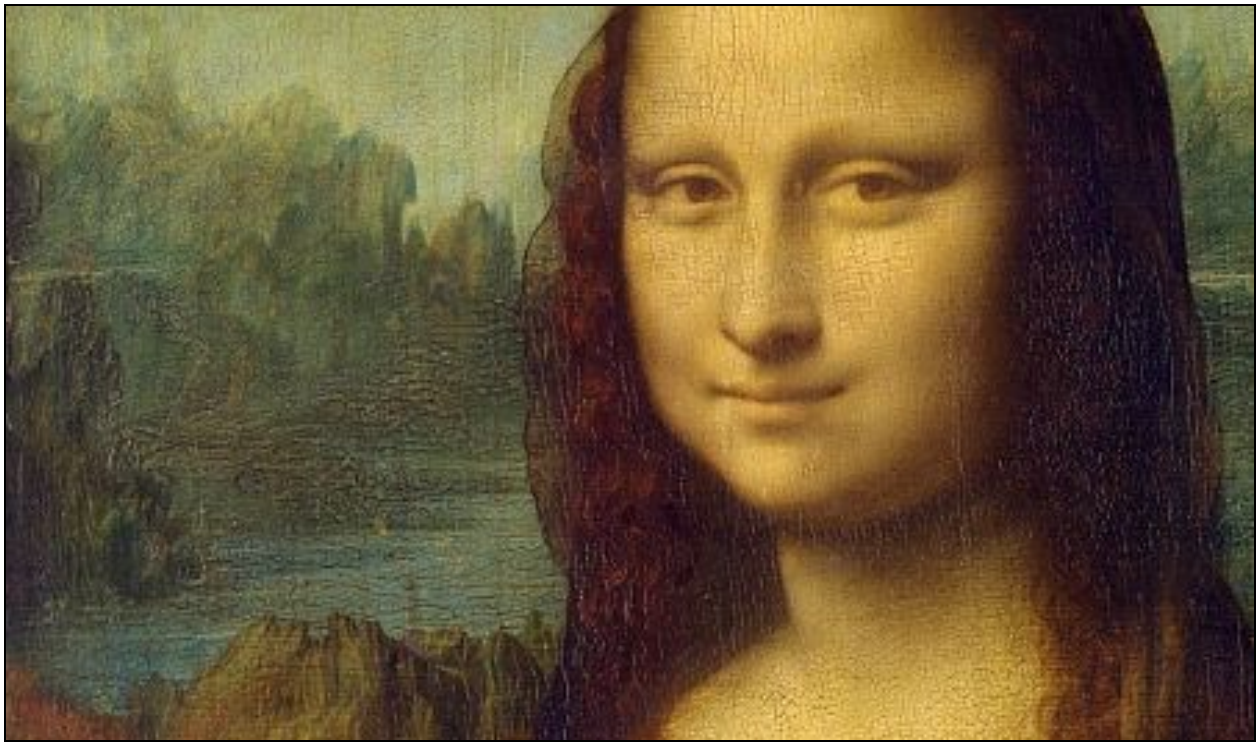
COMPARING MNB & SVM

Using Kaggle Sentiment Classification Dataset, Labeled by Amazon Turkers



Introduction

ARTIFICIAL INTELLIGENCE

What is 'Artificial Intelligence?' At the nexus of machines and humans is this strange hard-to-grasp, even-harder-to-quantify blanket term called Artificial Intelligence. Once a Hollywood blockbuster depicting one of the many strange futures and concepts that is Artificial Intelligence, it is now a silicon valley buzzword, like bitcoin or blockchain, used to excite stakeholders and artificially increase valuations.

In reality, Artificial Intelligence is considerably less glamorous. Artificial Intelligence is simply taking advantage of computers (no, not in that way iRobot enthusiasts) by utilizing their computing power across many different things that would be far too tedius (and error prone) for a human to do.

For example, let's say we want to know how the world feels about the President of the United States. In the olden days, before things like mass communication, computers and the internet, we might have to walk door to door, ring the doorbell, interview the inhabitants, take notes, and return to our university where we would manually sift through the notes pulling out words that might seem more "positive" or "negative" in nature. This could be manageable for one 2nd grader on his/her cul de sac, (I'd venture she'd disagree, though) but on a large scale, this is nearly impossible.

Let's pretend for a minute that we can magically snap our fingers and get a sentence from each person. If each person in the United States simply wrote one sentence about the President, we'd have over 300 million sentences to review. Even if it magically (call Hogwarts) took us one second to review and categorize each sentence, and we worked around the clock, it would take us over 9 years to do this -- and by then, we'd have a different president! Not only is this nearly impossible, it is quite ineffective. Computers, on the other hand, are quite effective at tasks like this.

Computers are absolutely amazing at menial tasks -- especially counting things. Computers are also very good at doing math quickly and efficiently with numbers too large even for our very expensive T.I. calculators. Computers have a lot of other skills but that is slightly (ahem, well) beyond the scope of this research paper. In short, Artificial Intelligence is using computers and machines to do things humans can't do as well, and often using things like counting and math to train computers to do even more amazing things.

ARTIFICIAL ARTIFICIAL INTELLIGENCE

What happens when we come across something a human still can do better than a machine? What happens when this task is something like "lie detection" where it's hard to quantify other than a "gut feeling?" How do we measure "gut feeling" and how can we train a computer on something so nebulous? What happens when we are woefully ill-equipped to teach a computer to do things because of our own inability to know exactly what minor calculations are going on inside our head that say, "Oh yes, this is sarcasm." Enter Amazon and their Mechanical Turk program, creatively touting their product as "Artificial" Artificial Intelligence. Backed by hundreds of thousands of workers ("turkers"), Amazon Mechanical Turk (AMT) farms out this "gut feeling" to humans for a meager sum, all the while collecting

the data with the ultimate goal of automating the turkers out of existence. However, until that day, humans are still behind AMT and they help those of us who are unfortunate enough to come to a research project with unlabeled data.

Analysis & Models

ABOUT THE DATA

Taken directly from the original data source:

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee [1]. In their work on sentiment treebanks, Socher et al. [2] used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. This competition presents a chance to benchmark your sentiment-analysis ideas on the Rotten Tomatoes dataset. You are asked to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. Obstacles like sentence negation, sarcasm, terseness, language ambiguity, and many others make this task very challenging.

The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data.

- train.tsv contains the phrases and their associated sentiment labels. We have additionally provided a SentenceId so that you can track which phrases belong to a single sentence.
- test.tsv contains just phrases. You must assign a sentiment label to each phrase.

The sentiment labels are:

- 0 - negative
- 1 - somewhat negative

- 2 - neutral
- 3 - somewhat positive
- 4 - positive

MODELS

MNB -- Multinomial Naive Bayes

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

SVM -- Support Vector Machines

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support-vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a (p-1)(p-1)-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum-margin classifier; or equivalently, the perceptron of optimal stability.

More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

Results

TASK 1

Build a unigram MNB model and a unigram SVM Model

VECTORIZOR: CountVectorizer

VECTORIZATION 1 | CountVectorizer, Binary, Unigram (No SW, min_df = 5)

MNB: 60.6%

```
*****CONFUSION MATRIX*****
[[  733  1264   817   106    11]
 [  602  4132  5411   649    30]
 [  246  2397 25756  3226   239]
 [   19   454  5580  6248   767]
 [    1    54   725  1972   985]]
*****CLASSIFICATION REPORT*****
              precision    recall  f1-score   support

           0       0.46      0.25      0.32      2931
           1       0.50      0.38      0.43     10824
           2       0.67      0.81      0.73     31864
           3       0.51      0.48      0.49     13068
           4       0.48      0.26      0.34      3737

    accuracy                           0.61     62424
   macro avg       0.53      0.44      0.47     62424
weighted avg       0.59      0.61      0.59     62424

*****SCORES*****
0.606401384083045
```

SVM: 62.4%

```
=====CONFUSION MATRIX=====
[[  913  1229   696    79    14]
 [  705  4094  5472   527    26]
 [  190  2111 27063  2324   176]
 [   33   394  6011  5568  1062]
 [    3    51   582  1775  1326]]
=====CLASSIFICATION REPORT=====
              precision    recall  f1-score   support

           0       0.50      0.31      0.38      2931
           1       0.52      0.38      0.44     10824
           2       0.68      0.85      0.76     31864
           3       0.54      0.43      0.48     13068
           4       0.51      0.35      0.42      3737
```

```
    accuracy                           0.62     62424
   macro avg       0.55      0.46      0.49     62424
weighted avg       0.60      0.62      0.60     62424

=====CONFIDENCE SCORES=====
[-1.04825431 -0.50286656  0.20910626 -0.97398088 -1.15145395]
=====SCORES=====
0.6241830065359477
```

BEST RESULT: 62.4% (SVM)

VECTORIZATION 2 | CountVectorizer, Unigram (No SW, min_df = 5)

MNB: 60.6%

```
*****CONFUSION MATRIX*****
[[  742  1276   797   105    11]
 [  614  4126  5397   655    32]
 [  248  2385 25756  3239   236]
 [   19   456  5570  6253   770]
 [    1    53   729  1977   977]]
*****CLASSIFICATION REPORT*****
              precision    recall  f1-score   support

           0       0.46      0.25      0.33      2931
           1       0.50      0.38      0.43     10824
           2       0.67      0.81      0.73     31864
           3       0.51      0.48      0.49     13068
           4       0.48      0.26      0.34      3737

    accuracy                           0.61     62424
   macro avg       0.52      0.44      0.47     62424
weighted avg       0.59      0.61      0.59     62424

*****SCORES*****
0.606401384083045
```

SVM: 62.3%

```
=====CONFUSION MATRIX=====
[[  918  1221   697    82    13]
 [  701  4080  5504   514    25]
```

```
 [  195  2106 27081  2310   172]
 [   34   396  6048  5533  1057]
 [    3    51   590  1772  1321]]
=====CLASSIFICATION REPORT=====
              precision    recall  f1-score   support

           0       0.50      0.31      0.38      2931
           1       0.52      0.38      0.44     10824
           2       0.68      0.85      0.75     31864
           3       0.54      0.42      0.48     13068
           4       0.51      0.35      0.42      3737

    accuracy                           0.62     62424
   macro avg       0.55      0.46      0.49     62424
weighted avg       0.60      0.62      0.60     62424

=====CONFIDENCE SCORES=====
[-1.01718415 -0.50760036  0.22331226 -0.97514722 -1.24718831]
=====SCORES=====
0.6236864026656415
```

BEST RESULT: 62.3% (SVM) -- Slightly worse than Vec 1

VECTORIZATION 3 | CountVectorizer, Bigram (No SW, min_df = 5)

MNB: 59.7

```
*****CONFUSION MATRIX*****
[[  867  1253   725    69    17]
 [  786  4440  4943   609    46]
 [  459  2961 24437  3600   407]
 [   41   513  5082  6375  1057]
 [    6    46   602  1911  1172]]
*****CLASSIFICATION REPORT*****
              precision    recall  f1-score   support

           0       0.40      0.30      0.34      2931
           1       0.48      0.41      0.44     10824
           2       0.68      0.77      0.72     31864
           3       0.51      0.49      0.50     13068
           4       0.43      0.31      0.36      3737
```

```
    accuracy                           0.60     62424
   macro avg       0.50      0.45      0.47     62424
weighted avg       0.58      0.60      0.59     62424

*****SCORES*****
0.5973824170190952
```

SVM: 63%

```
=====CONFUSION MATRIX=====
[[ 1039  1276   542    63    11]
 [  864  4555  4911   457    37]
 [  252  2470 26246  2700   196]
 [   28   358  5383  6034  1265]
 [    5    27   452  1794  1459]]
=====CLASSIFICATION REPORT=====
              precision    recall  f1-score   support

           0       0.47      0.35      0.41      2931
           1       0.52      0.42      0.47     10824
           2       0.70      0.82      0.76     31864
           3       0.55      0.46      0.50     13068
           4       0.49      0.39      0.44      3737

    accuracy                           0.63     62424
   macro avg       0.55      0.49      0.51     62424
weighted avg       0.61      0.63      0.62     62424

=====CONFIDENCE SCORES=====
[-1.35329509 -0.56433734  0.50417972 -0.98434221 -1.14487822]
=====SCORES=====
0.6300941945405614
```

BEST RESULT: 63% (SVM) -- Slightly better than Vec 1 & Vec2

VECTORIZOR: TfidfVectorizer

VECTORIZATION 4| TfidfVectorizer, Unigram (No SW, min_df = 5)

MNB: 58.4%

```
*****CONFUSION MATRIX*****
[[  107  1144  1613    67     0]
 [   61  2580  7821   361     1]
 [   19  1168 28673  1987    17]
 [    0   147  7942  4883    96]
 [    0    11  1374  2164   188]]
*****CLASSIFICATION REPORT*****
              precision    recall  f1-score   support

           0       0.57      0.04      0.07      2931
           1       0.51      0.24      0.33     10824
           2       0.60      0.90      0.72     31864
           3       0.52      0.37      0.43     13068
           4       0.62      0.05      0.09      3737

    accuracy                           0.58     62424
   macro avg       0.57      0.32      0.33     62424
weighted avg       0.57      0.58      0.53     62424

*****SCORES*****
0.5836056644880174
```

SVM: 62.5%

```
=====CONFUSION MATRIX=====
[[  795  1387   624   117     8]
 [  589  4336  5245   629    25]
 [  163  2299 26557  2684   161]
 [   24   408  5604  6220   812]
 [    2    40   551  2010  1134]]
=====CLASSIFICATION REPORT=====
              precision    recall  f1-score   support

           0       0.51      0.27      0.35      2931
           1       0.51      0.40      0.45     10824
           2       0.69      0.83      0.75     31864
           3       0.53      0.48      0.50     13068
           4       0.53      0.30      0.39      3737
```

```
    accuracy                           0.63     62424
   macro avg       0.55      0.46      0.49     62424
weighted avg       0.61      0.63      0.61     62424

=====CONFIDENCE SCORES=====
[-1.01488208 -0.38030889  0.16542161 -0.97048325 -1.23292618]
=====SCORES=====
0.6254325259515571
```

BEST RESULT: 62.5% (SVM) -- Worse than Vec 3

VECTORIZATION 5| TfidfVectorizer, Unigram (No SW, min_df = 5, max_df)

MNB: 58.4%

```
*****CONFUSION MATRIX*****
[[  107  1144  1613    67     0]
 [   61  2580  7821   361     1]
 [   19  1168 28673  1987    17]
 [    0   147  7942  4883    96]
 [    0    11  1374  2164   188]]
*****CLASSIFICATION REPORT*****
              precision    recall  f1-score   support

           0       0.57      0.04      0.07      2931
           1       0.51      0.24      0.33     10824
           2       0.60      0.90      0.72     31864
           3       0.52      0.37      0.43     13068
           4       0.62      0.05      0.09      3737

    accuracy                           0.58     62424
   macro avg       0.57      0.32      0.33     62424
weighted avg       0.57      0.58      0.53     62424

*****SCORES*****
0.5836056644880174
```

SVM: 62.5%

```
=====CONFUSION MATRIX=====
[[  795  1387   624   117     8]
 [  589  4336  5245   629    25]
```

```
 [  163  2299 26557  2684   161]
 [   24   408  5604  6220   812]
 [    2    40   551  2010  1134]]
=====CLASSIFICATION REPORT=====
              precision    recall  f1-score   support

           0       0.51      0.27      0.35      2931
           1       0.51      0.40      0.45     10824
           2       0.69      0.83      0.75     31864
           3       0.53      0.48      0.50     13068
           4       0.53      0.30      0.39      3737

    accuracy                           0.63     62424
   macro avg       0.55      0.46      0.49     62424
weighted avg       0.61      0.63      0.61     62424

=====CONFIDENCE SCORES=====
[-1.01488249 -0.38032514  0.16541625 -0.97048002 -1.23292607]
=====SCORES=====
0.6254325259515571
```

BEST RESULT: 62.5% (SVM) -- Worse than Vec 3, Same as Vec 4

VECTORIZATION 6| TfidfVectorizer, Bigram (No SW, min_df = 5, max_df)

MNB: 59.4%

```
*****CONFUSION MATRIX*****
[[  179  1186  1513    52     1]
 [   77  2868  7598   279     2]
 [   18  1242 28695  1897    12]
 [    1   140  7680  5128   119]
 [    0    18  1326  2127   266]]
*****CLASSIFICATION REPORT*****
              precision    recall  f1-score   support

           0       0.65      0.06      0.11      2931
           1       0.53      0.26      0.35     10824
           2       0.61      0.90      0.73     31864
           3       0.54      0.39      0.45     13068
           4       0.67      0.07      0.13      3737
```

```
    accuracy                           0.59     62424
   macro avg       0.60      0.34      0.36     62424
weighted avg       0.59      0.59      0.54     62424

*****SCORES*****
0.5948993976675637
```

SVM: 62.5%

```
=====CONFUSION MATRIX=====
[[  916  1373   565    69     8]
 [  696  4666  4947   493    22]
 [  217  2507 26156  2827   157]
 [   25   364  5343  6334  1002]
 [    5    32   475  1962  1263]]
=====CLASSIFICATION REPORT=====
              precision    recall  f1-score   support

           0       0.49      0.31      0.38      2931
           1       0.52      0.43      0.47     10824
           2       0.70      0.82      0.75     31864
           3       0.54      0.48      0.51     13068
           4       0.52      0.34      0.41      3737

    accuracy                           0.63     62424
   macro avg       0.55      0.48      0.51     62424
weighted avg       0.61      0.63      0.62     62424

=====CONFIDENCE SCORES=====
[-1.17972911 -0.41383963  0.29126027 -0.87403664 -1.04112626]
=====SCORES=====
0.6301262334999359
```

BEST RESULT: 63% (SVM) Same as Vec 3, CountVectorizer with Bigrams

TASK 2

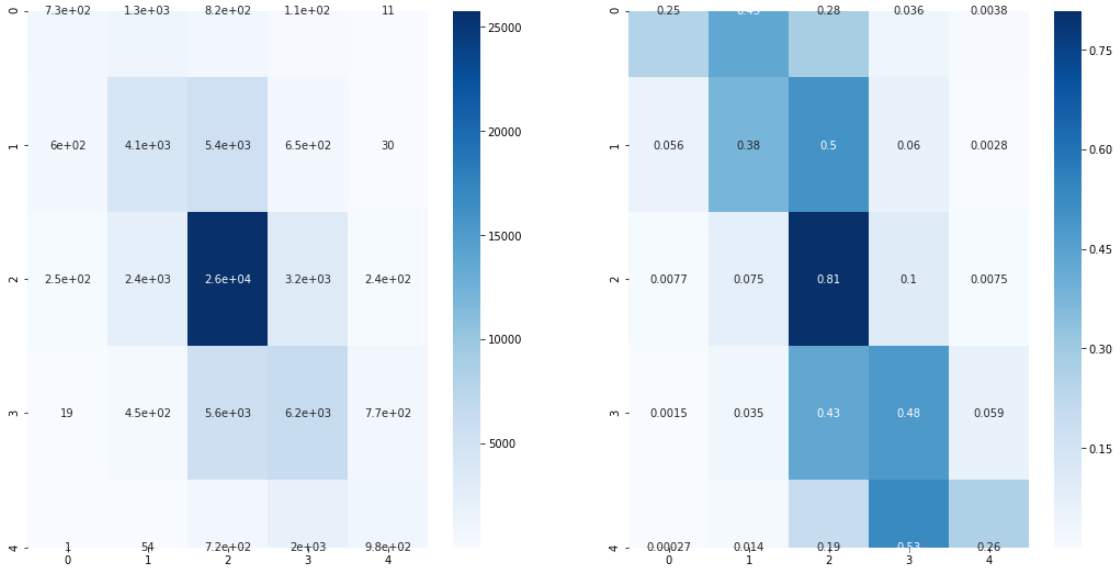
`Vectorizer Settings for 0_CountVectorizer`

<Figure size 432x288 with 0 Axes>

index	0
analyzer	word
binary	True
decode_error	strict
dtype	<class 'numpy.int64'>
encoding	latin-1
input	content
lowercase	True
max_df	1.0
max_features	
min_df	5
ngram_range	(1, 1)
preprocessor	
stop_words	english
strip_accents	
token_pattern	(?u)\b\w\w+\b
tokenizer	
vocabulary	

png

```
----------------------------------------
##MNB
----------------------------------------
Confusion Matrices: Non-normalized and Normalized
```



png

==========

BY SENTIMENT

==========

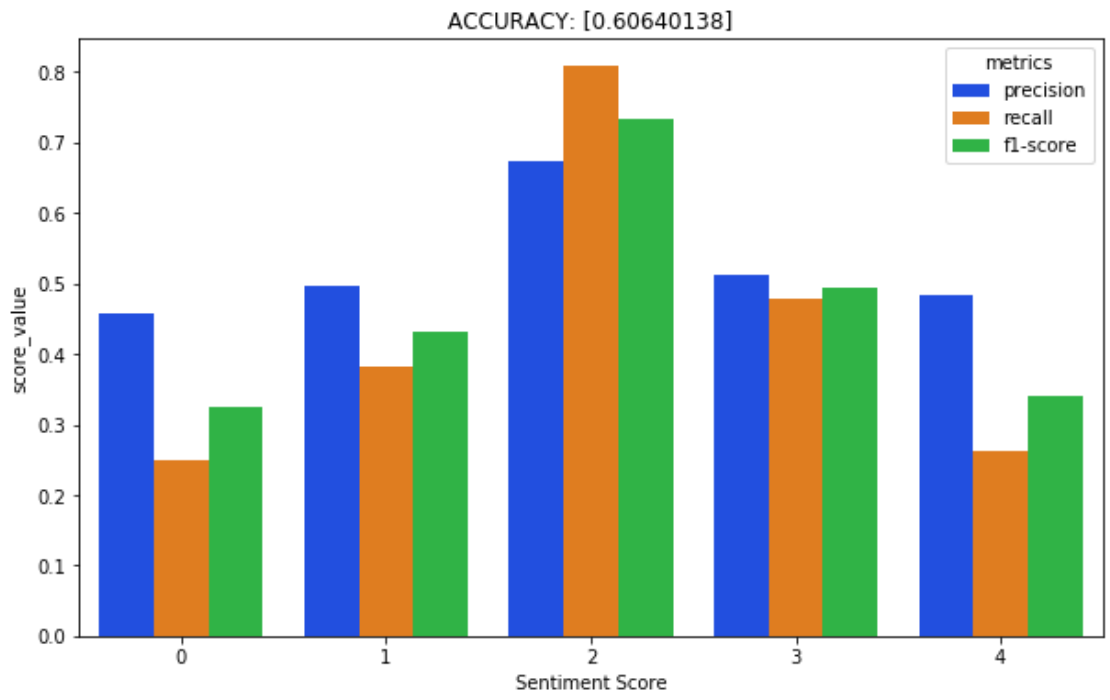
	0	1	2	3	4
precision	0.46	0.50	0.67	0.51	0.48
recall	0.25	0.38	0.81	0.48	0.26
f1-score	0.32	0.43	0.73	0.49	0.34
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

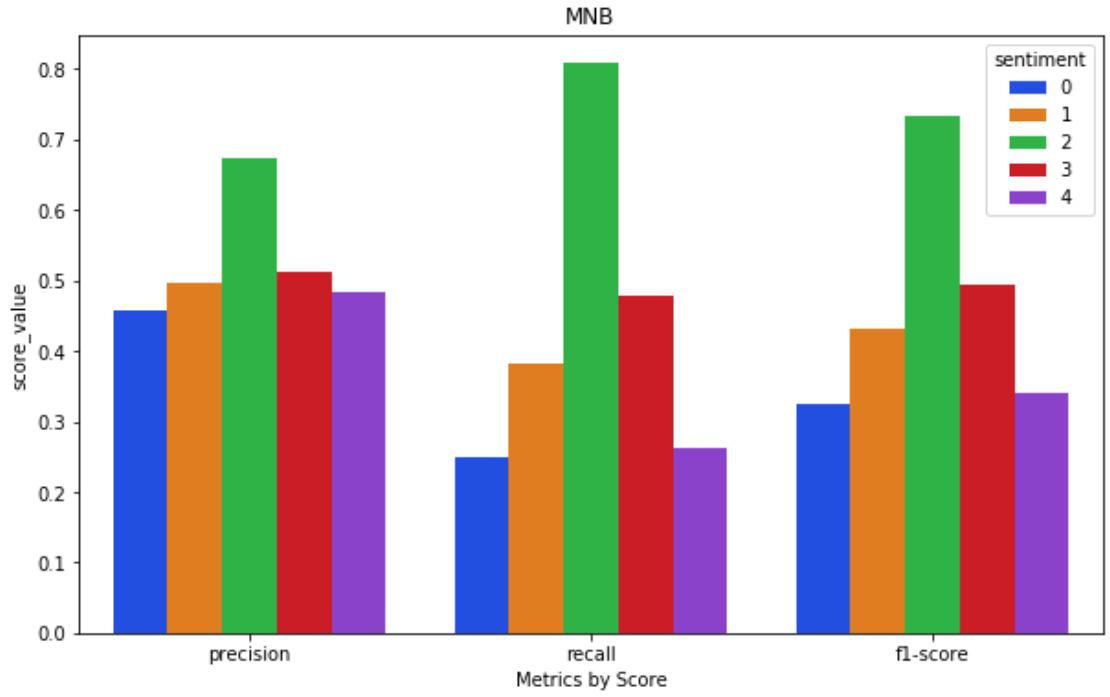
BY PERFORMANCE

==========

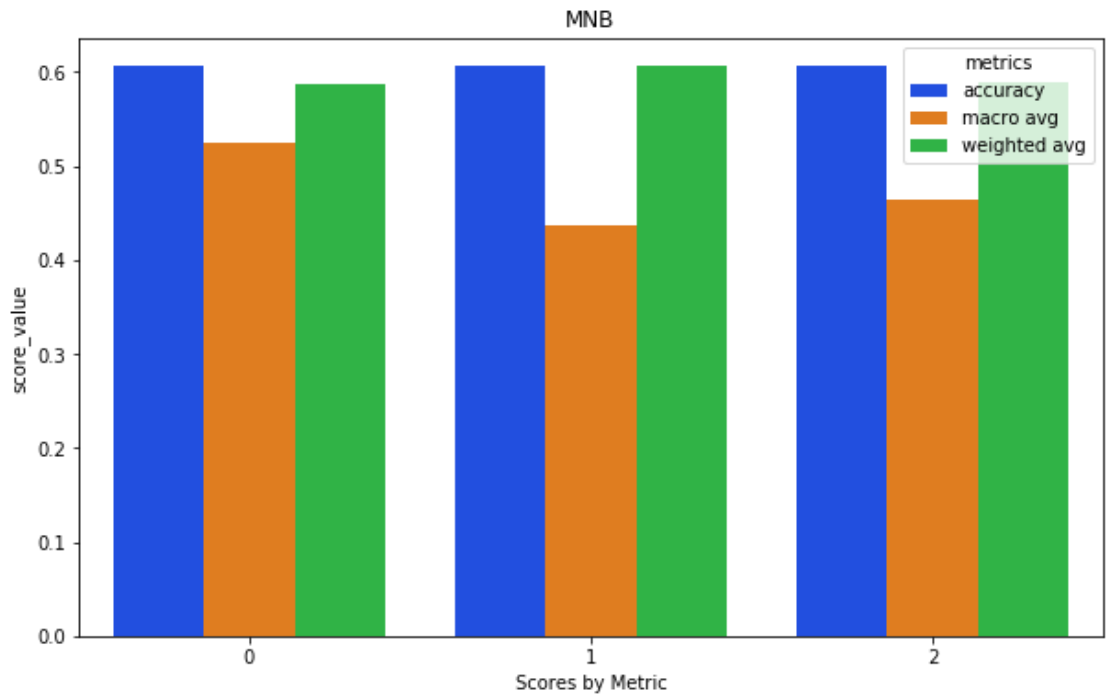
	accuracy	macro avg	weighted avg
precision	0.61	0.53	0.59
recall	0.61	0.44	0.61
f1-score	0.61	0.47	0.59
support	0.61	62424.00	62424.00


ACCURACY: [0.60640138]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

png


MNB
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

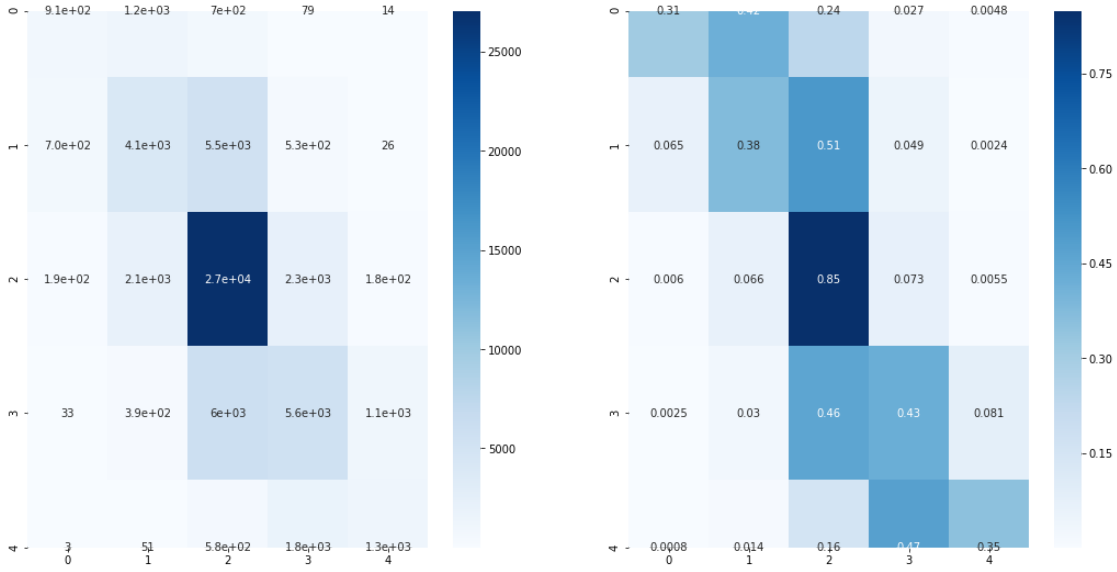
png



png

##SVM

Confusion Matrices: Non-normalized and Normalized



png

==========

BY SENTIMENT

==========

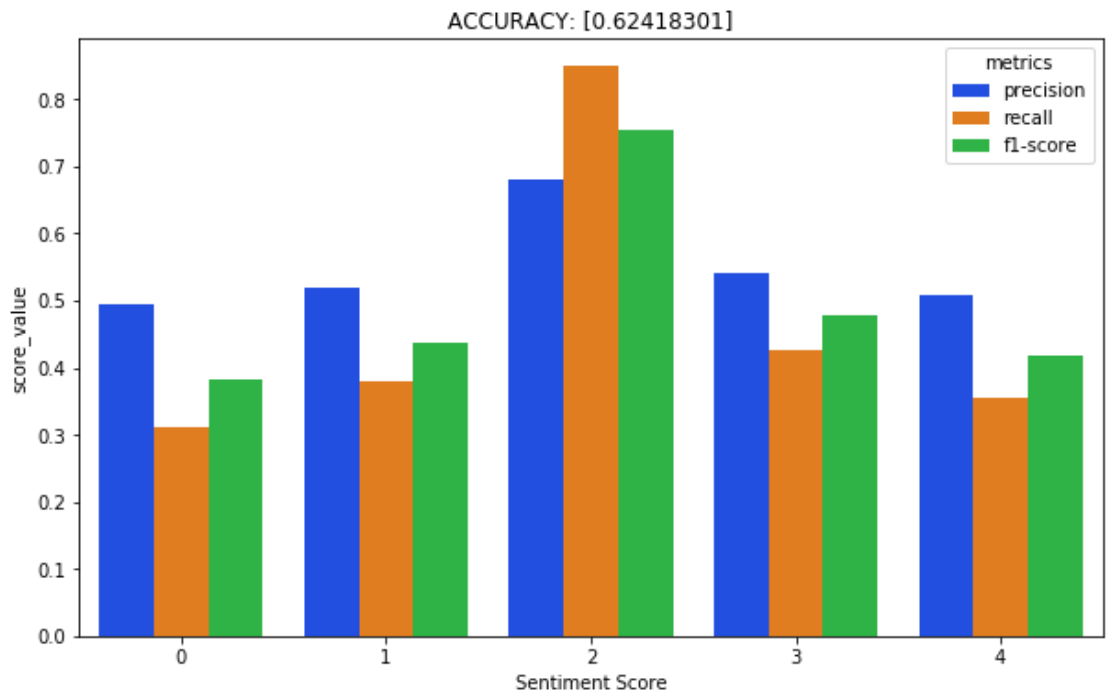
	0	1	2	3	4
precision	0.50	0.52	0.68	0.54	0.51
recall	0.31	0.38	0.85	0.43	0.35
f1-score	0.38	0.44	0.76	0.48	0.42
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

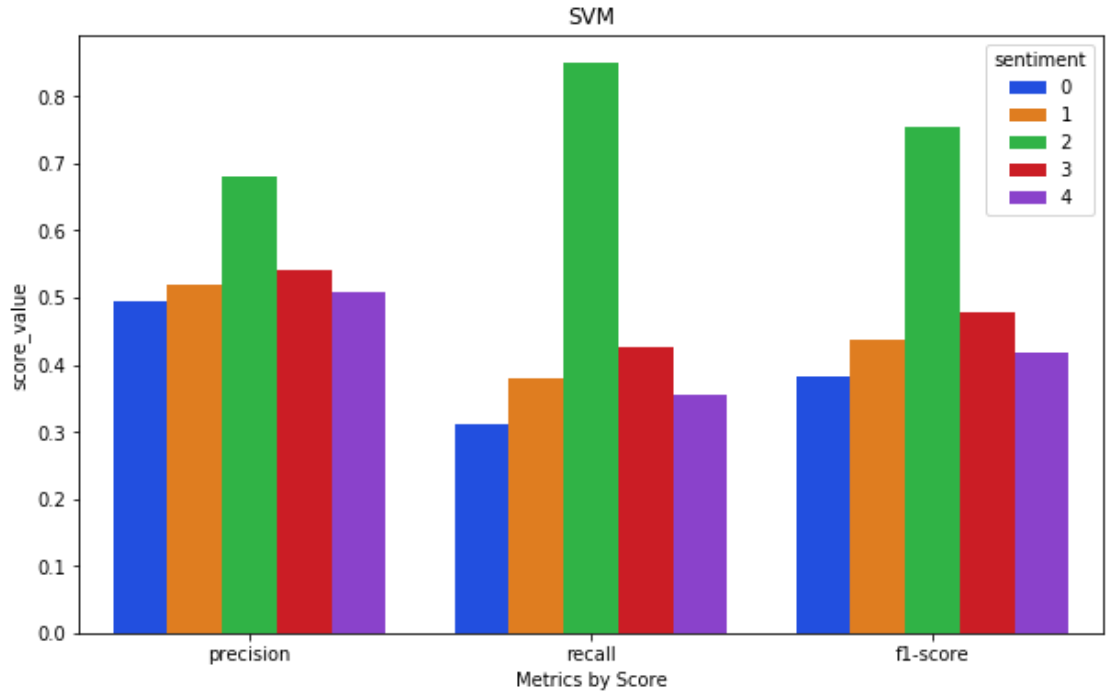
BY PERFORMANCE

==========

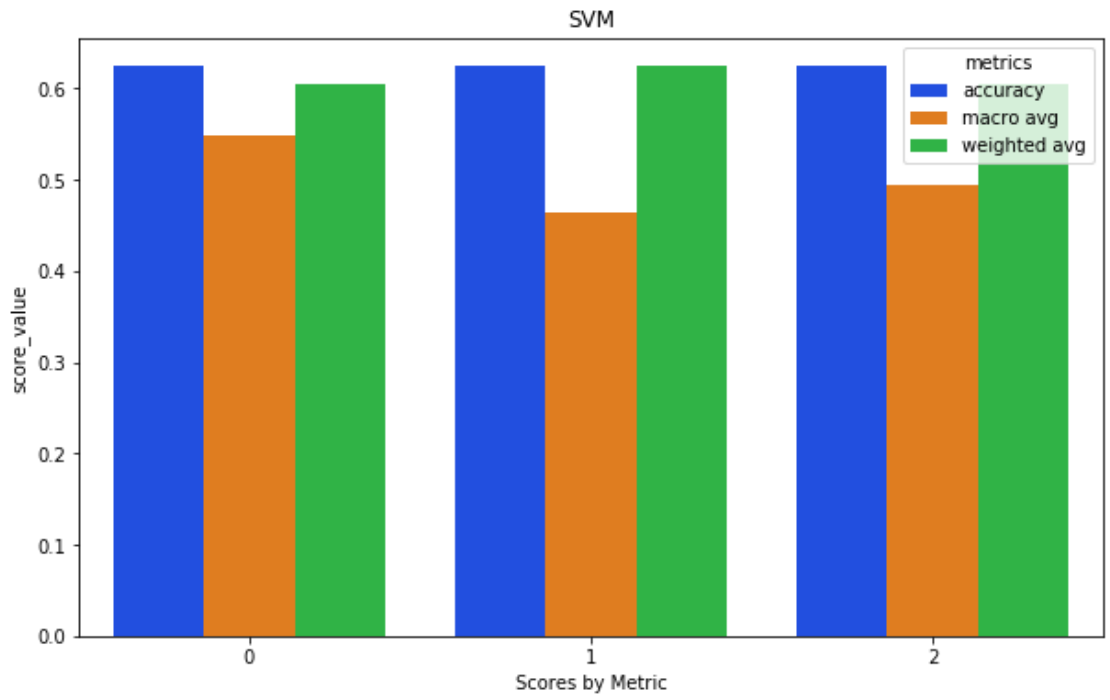
	accuracy	macro avg	weighted avg
precision	0.62	0.55	0.60
recall	0.62	0.46	0.62
f1-score	0.62	0.49	0.60
support	0.62	62424.00	62424.00


ACCURACY: [0.62418301]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

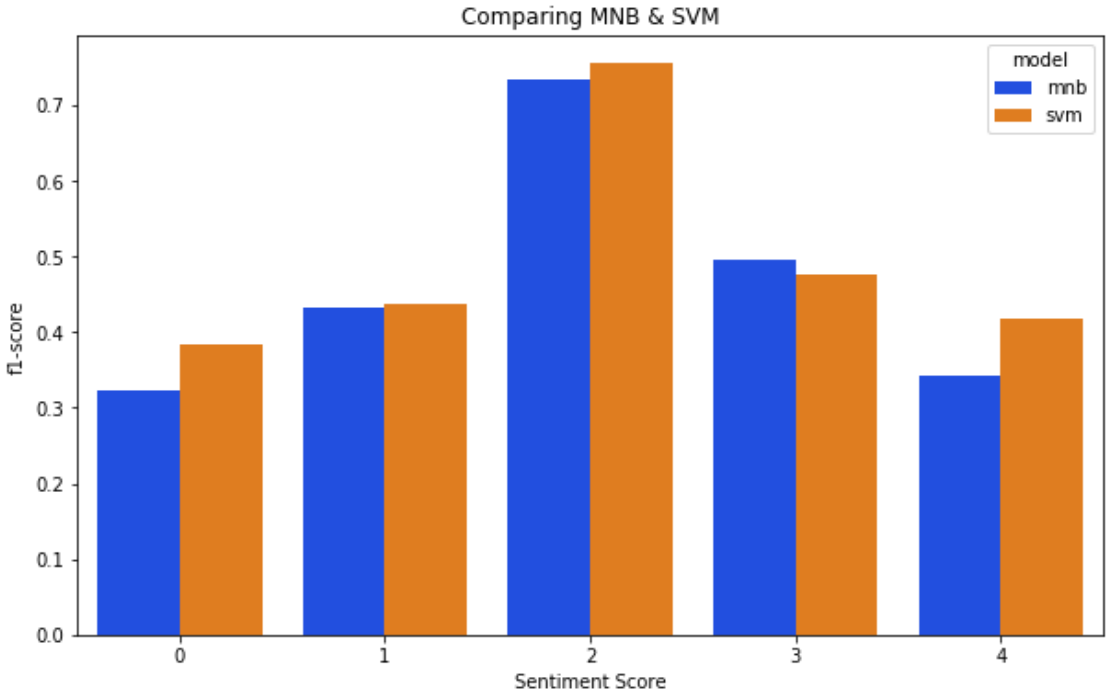
png


SVM
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

png



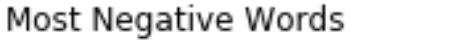
png



png

MNB FEATURES**********

```
<Figure size 432x288 with 0 Axes>
```



0	1
-5.946405317563997	time
-5.935710028447249	minutes
-5.925127919116712	characters
-5.925127919116712	story
-5.90429383221387	comedy
-5.6998813176057235	just
-5.195801081979265	like
-5.071833039257134	bad
-4.847793028895888	film
-4.322025825131698	movie

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-10.479004810717253	102
-10.479004810717253	10th
-10.479004810717253	127
-10.479004810717253	13th
-10.479004810717253	14
-10.479004810717253	16
-10.479004810717253	163
-10.479004810717253	168
-10.479004810717253	170
-10.479004810717253	1790

png

SVM FEATURES******************************

<Figure size 432x288 with 0 Axes>

Most Negative Words

0	1
1.6275629062323609	cesspool
1.663771776080994	pompous
1.691827980110218	stinks
1.698773923968737	distasteful
1.7121676106191022	unwatchable
1.7288294879442054	disappointment
1.7592903767925694	unbearable
1.808540820054038	stinker
1.8268010314534087	worthless
1.8341326994846554	disgusting

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-1.8437631760358908	hawke
-1.6993727122424316	collar
-1.697685877974192	giddy
-1.591086365564618	swimfan
-1.5714608901242135	blue
-1.4870189803119356	dogtown
-1.4322895156501692	victim
-1.4158506845365442	joan
-1.413733890385405	won
-1.4045010207220074	innocence

png

```
Vectorizer Settings for 1_CountVectorizer
```

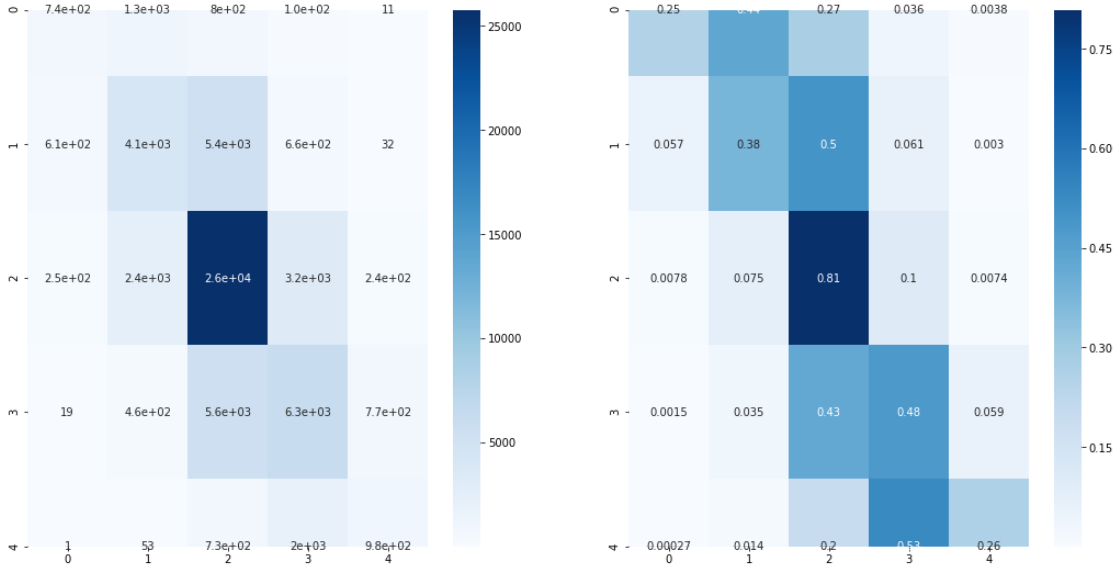
```
<Figure size 432x288 with 0 Axes>
```

index	0
analyzer	word
binary	False
decode_error	strict
dtype	<class 'numpy.int64'>
encoding	latin-1
input	content
lowercase	True
max_df	1.0
max_features	
min_df	5
ngram_range	(1, 1)
preprocessor	
stop_words	english
strip_accents	
token_pattern	(?u)\b\w\w+\b
tokenizer	
vocabulary	

png

##MNB

Confusion Matrices: Non-normalized and Normalized



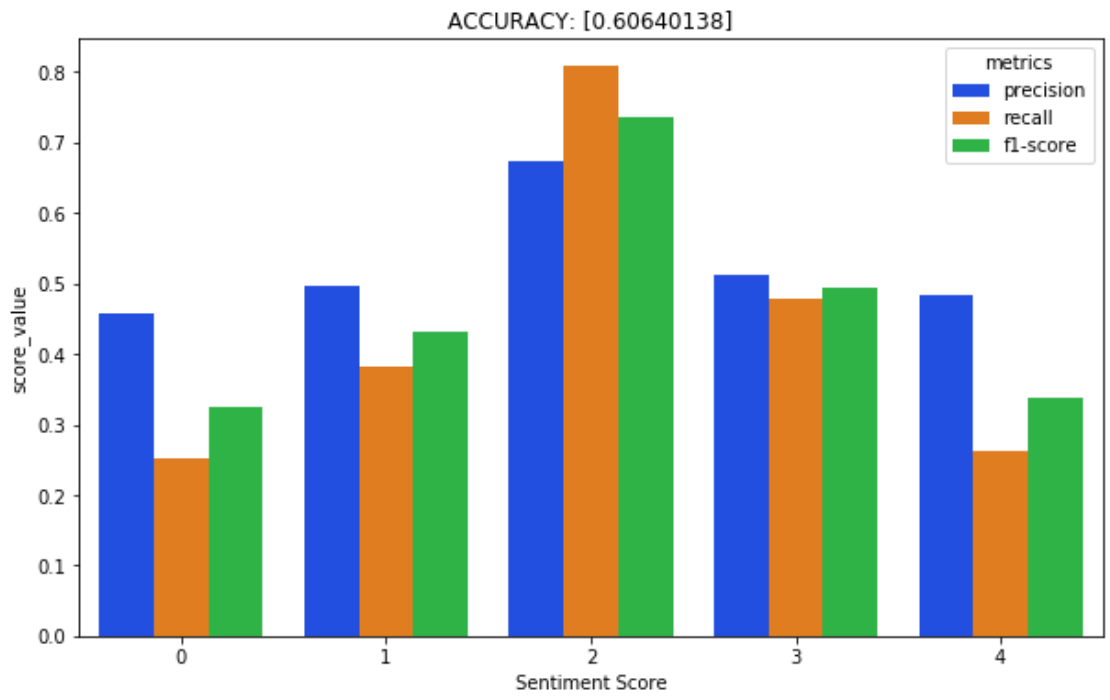
png

==========
BY SENTIMENT
==========

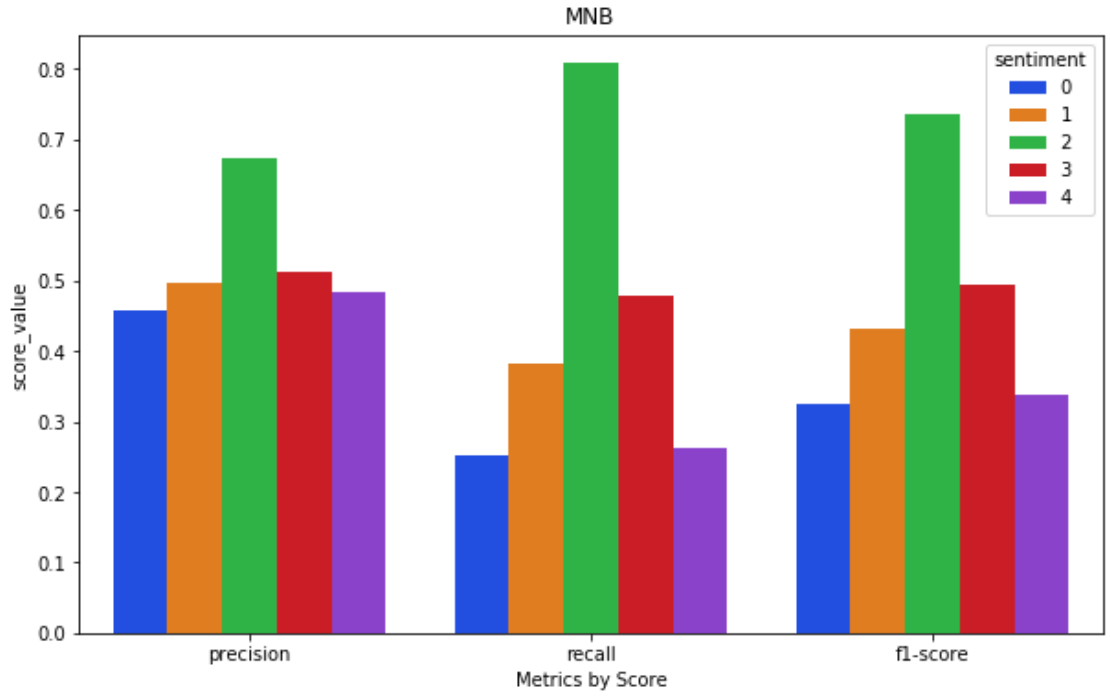
	0	1	2	3	4
precision	0.46	0.50	0.67	0.51	0.48
recall	0.25	0.38	0.81	0.48	0.26
f1-score	0.33	0.43	0.73	0.49	0.34
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========
BY PERFORMANCE
==========

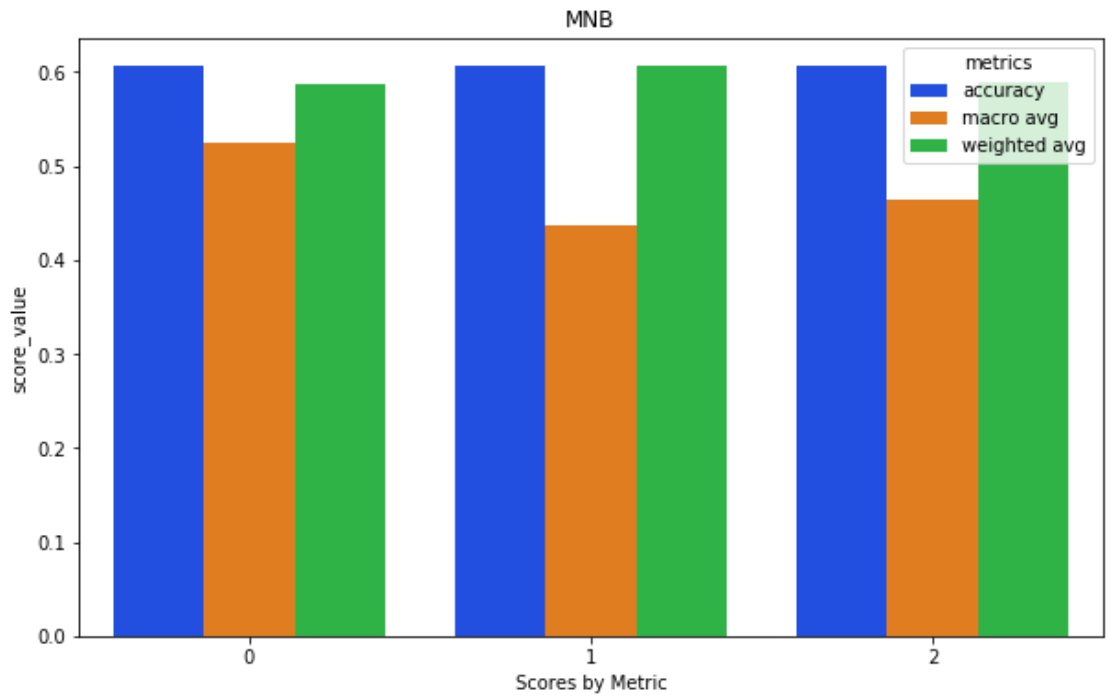
	accuracy	macro avg	weighted avg
precision	0.61	0.52	0.59
recall	0.61	0.44	0.61
f1-score	0.61	0.47	0.59
support	0.61	62424.00	62424.00


ACCURACY: [0.60640138]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

png


MNB
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

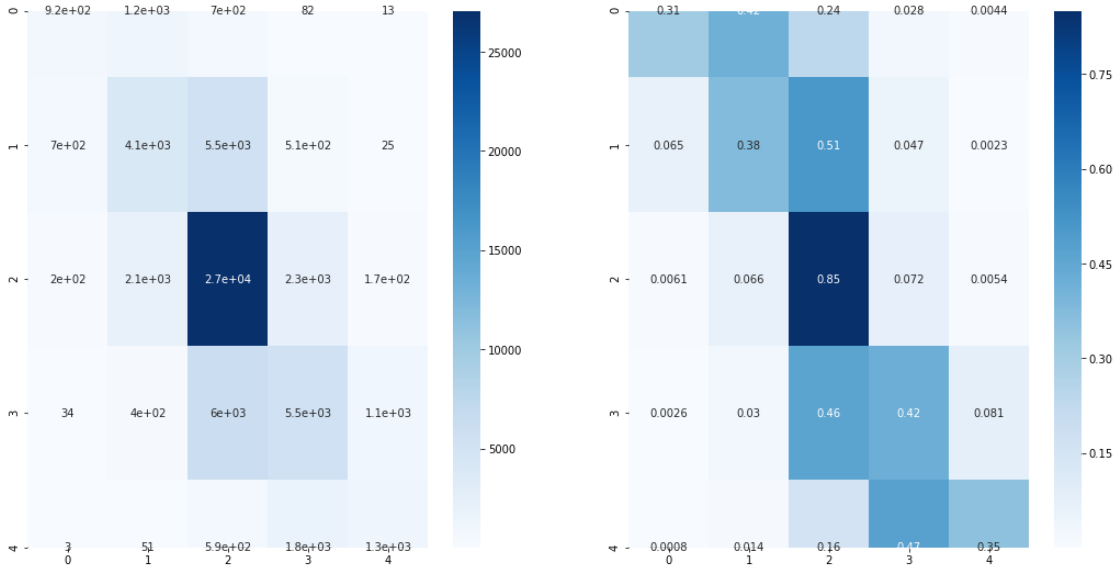
png



png

##SVM

Confusion Matrices: Non-normalized and Normalized



png

==========

BY SENTIMENT

==========

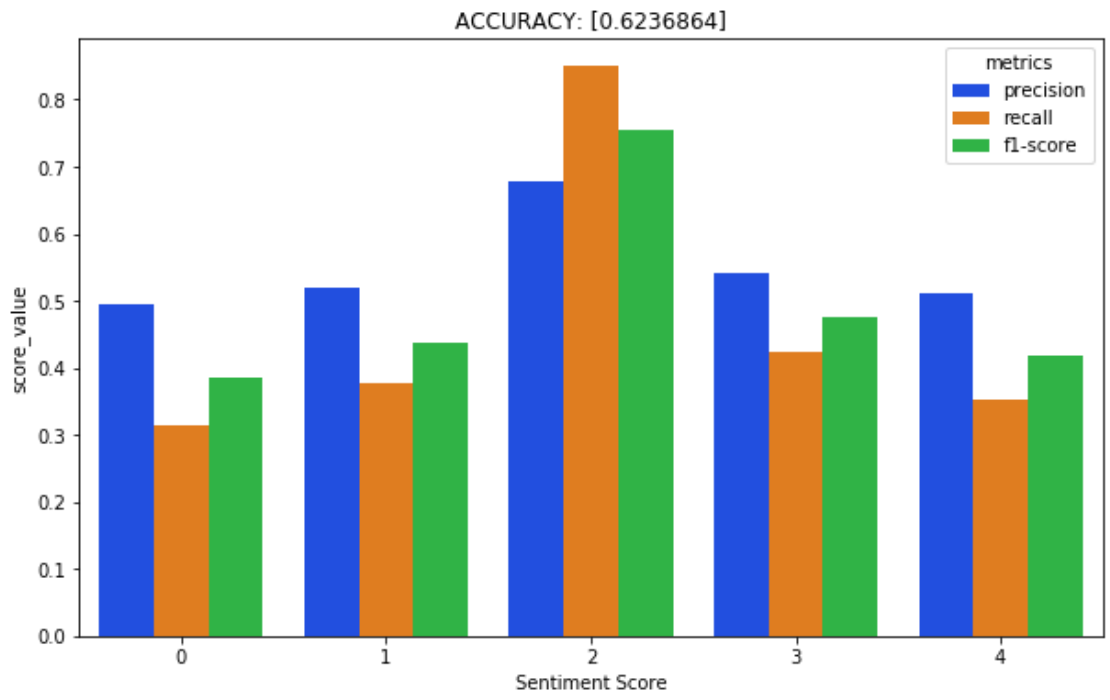
	0	1	2	3	4
precision	0.50	0.52	0.68	0.54	0.51
recall	0.31	0.38	0.85	0.42	0.35
f1-score	0.38	0.44	0.75	0.48	0.42
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

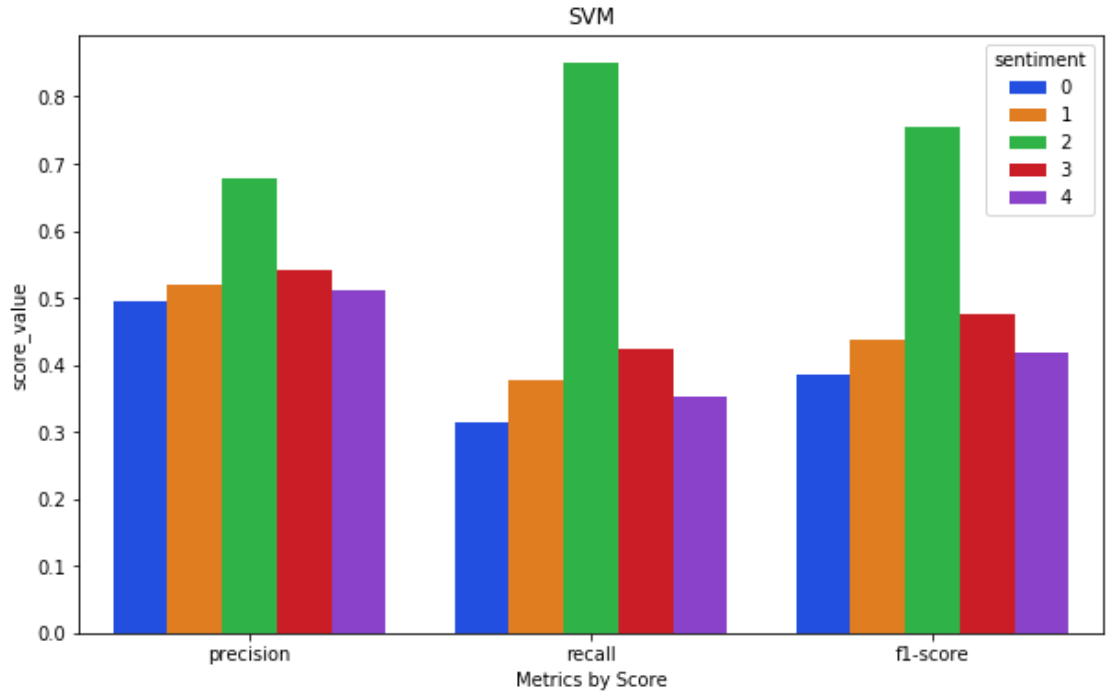
BY PERFORMANCE

==========

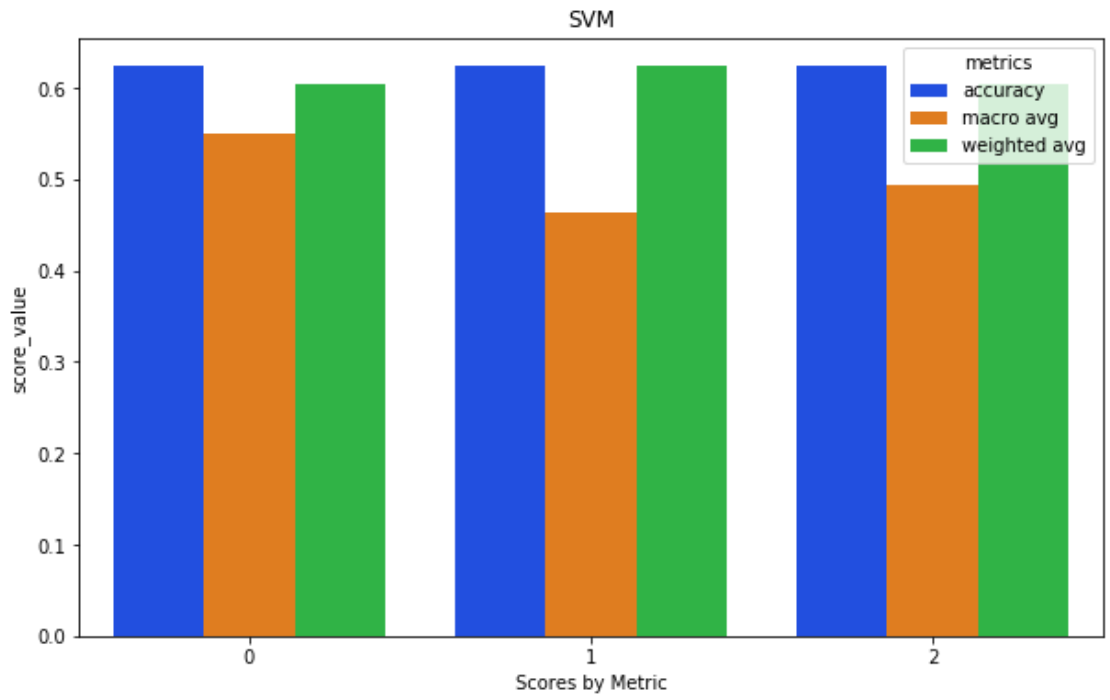
	accuracy	macro avg	weighted avg
precision	0.62	0.55	0.60
recall	0.62	0.46	0.62
f1-score	0.62	0.49	0.60
support	0.62	62424.00	62424.00


ACCURACY: [0.6236864]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

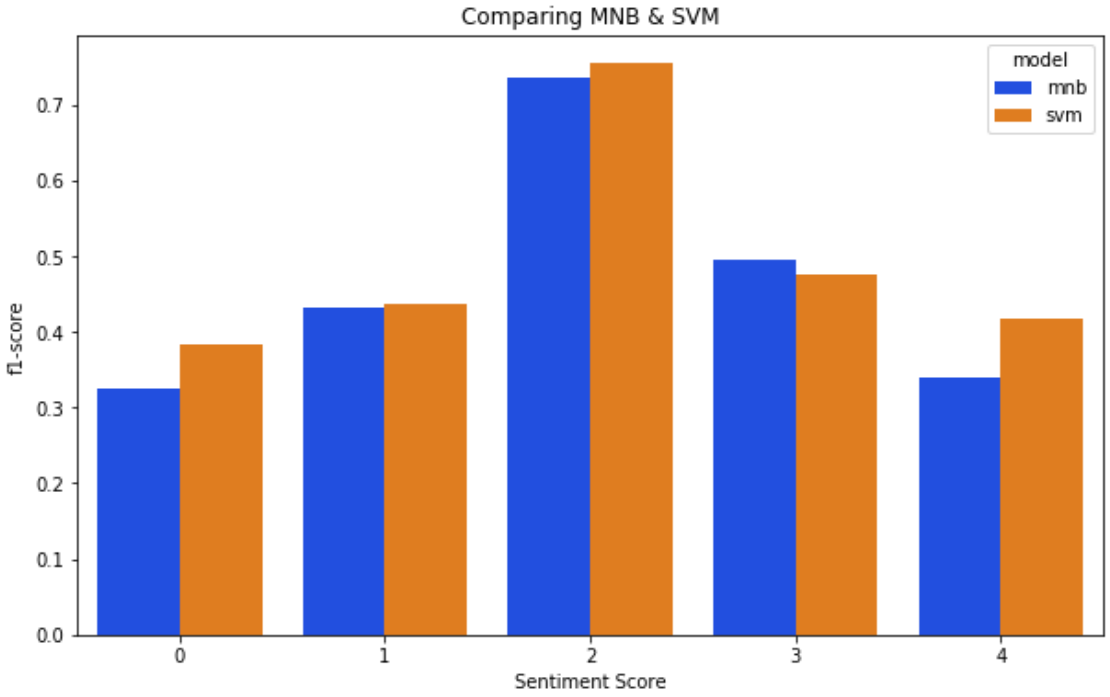
png


SVM
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

png



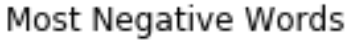
png



png

MNB FEATURES**********

```
<Figure size 432x288 with 0 Axes>
```



0	1
-5.941598005980322	time
-5.931015896649785	characters
-5.92054459678249	minutes
-5.92054459678249	story
-5.910181809746943	comedy
-5.689102242653584	just
-5.137785257532857	like
-4.975504451622348	bad
-4.832403607981675	film
-4.3215779842156845	movie

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-10.484892788250326	102
-10.484892788250326	10th
-10.484892788250326	127
-10.484892788250326	13th
-10.484892788250326	14
-10.484892788250326	16
-10.484892788250326	163
-10.484892788250326	168
-10.484892788250326	170
-10.484892788250326	1790

png

SVM FEATURES******************************

<Figure size 432x288 with 0 Axes>

Most Negative Words

0	1
1.621610015629506	cesspool
1.6484880696791182	disappointment
1.65924940316425	pompous
1.668369676175682	stinks
1.6927739430823703	distasteful
1.6955905376233327	unwatchable
1.752639842504262	unbearable
1.7873567328215978	stinker
1.8228705903573794	disgusting
1.8233055655862724	worthless

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-1.8329268977779145	hawke
-1.7372807294588553	giddy
-1.6832954335298385	collar
-1.5847291818782818	swimfan
-1.5720765506163714	blue
-1.4801113579603056	dogtown
-1.4138361240218211	clamoring
-1.4093532960690858	joan
-1.3918160207941686	victim
-1.3400001729841238	compulsively

png

```
Vectorizer Settings for 2_CountVectorizer
```

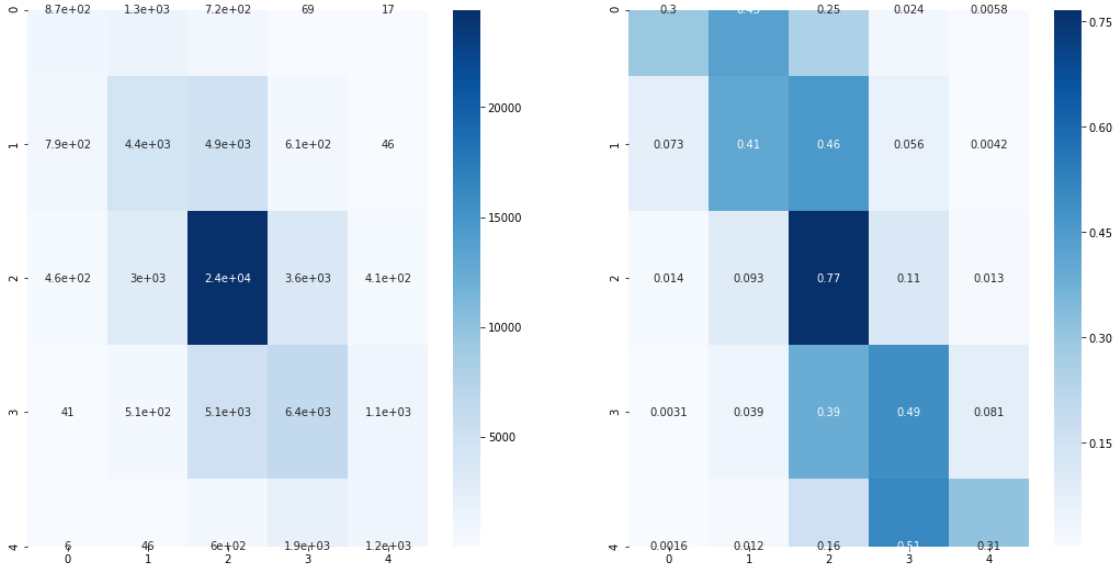
```
<Figure size 432x288 with 0 Axes>
```

index	0
analyzer	word
binary	False
decode_error	strict
dtype	<class 'numpy.int64'>
encoding	latin-1
input	content
lowercase	True
max_df	1.0
max_features	
min_df	5
ngram_range	(1, 2)
preprocessor	
stop_words	english
strip_accents	
token_pattern	(?u)\b\w\w+\b
tokenizer	
vocabulary	

png

##MNB

Confusion Matrices: Non-normalized and Normalized



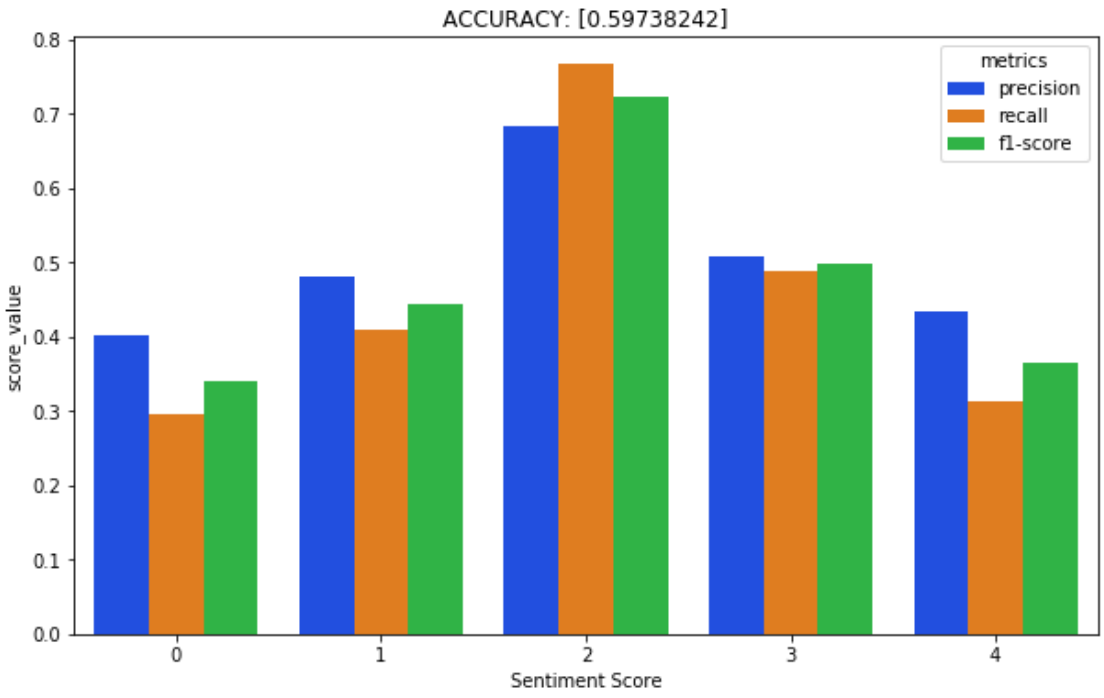
png

==========
BY SENTIMENT
==========

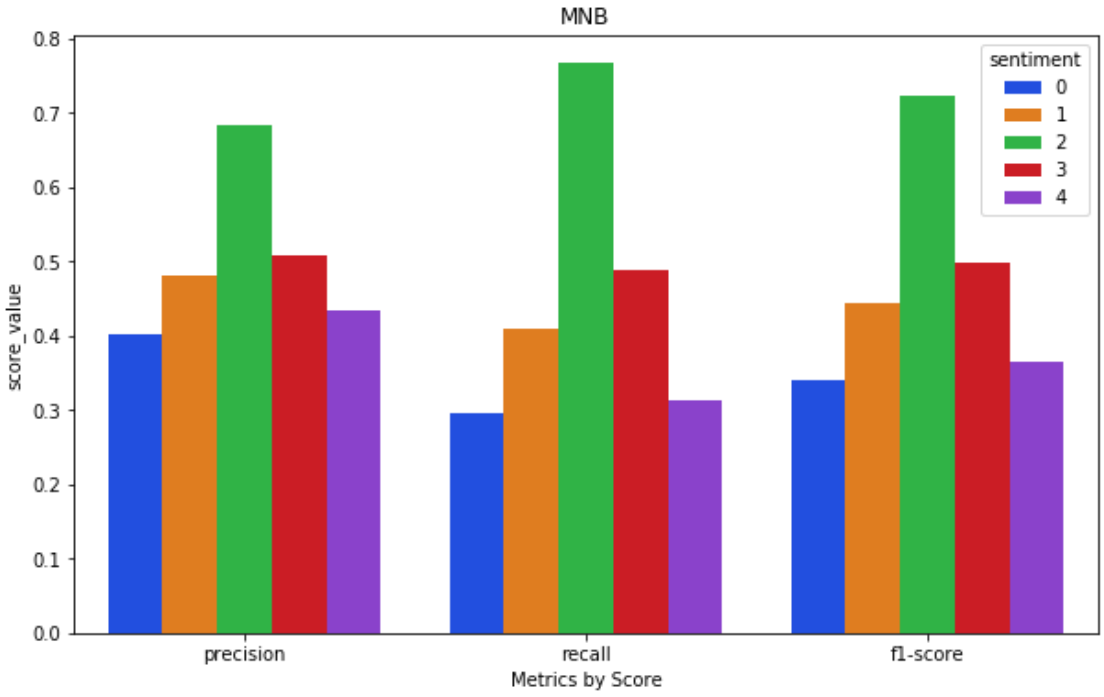
	0	1	2	3	4
precision	0.40	0.48	0.68	0.51	0.43
recall	0.30	0.41	0.77	0.49	0.31
f1-score	0.34	0.44	0.72	0.50	0.36
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========
BY PERFORMANCE
==========

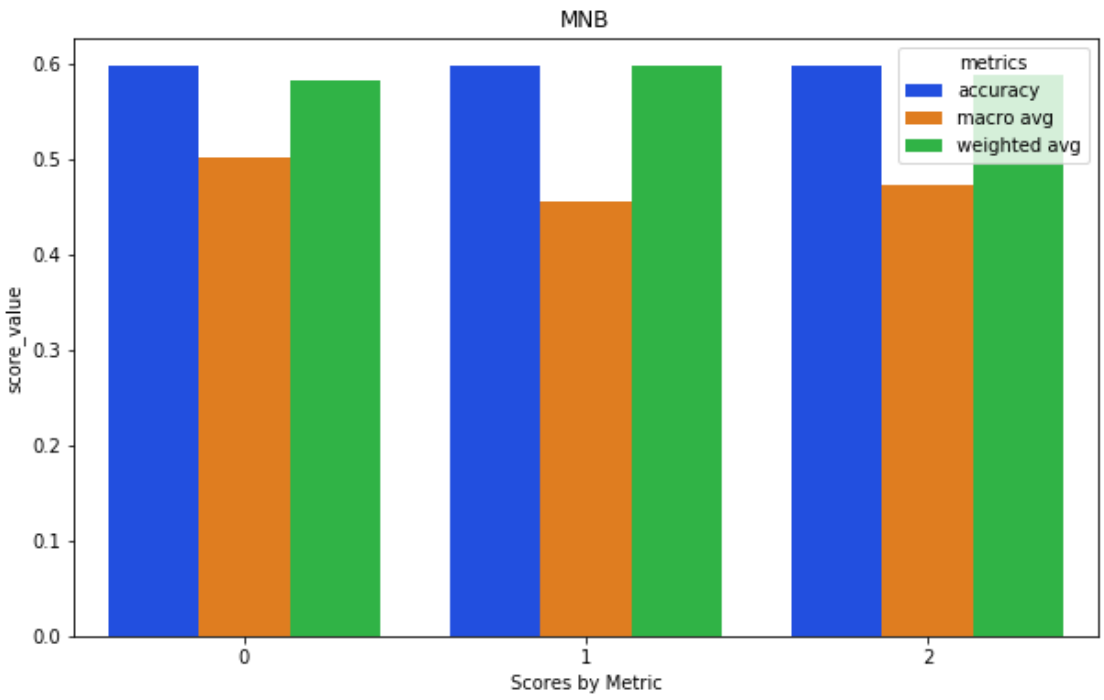
	accuracy	macro avg	weighted avg
precision	0.60	0.50	0.58
recall	0.60	0.45	0.60
f1-score	0.60	0.47	0.59
support	0.60	62424.00	62424.00


ACCURACY: [0.59738242]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

png


MNB
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

png

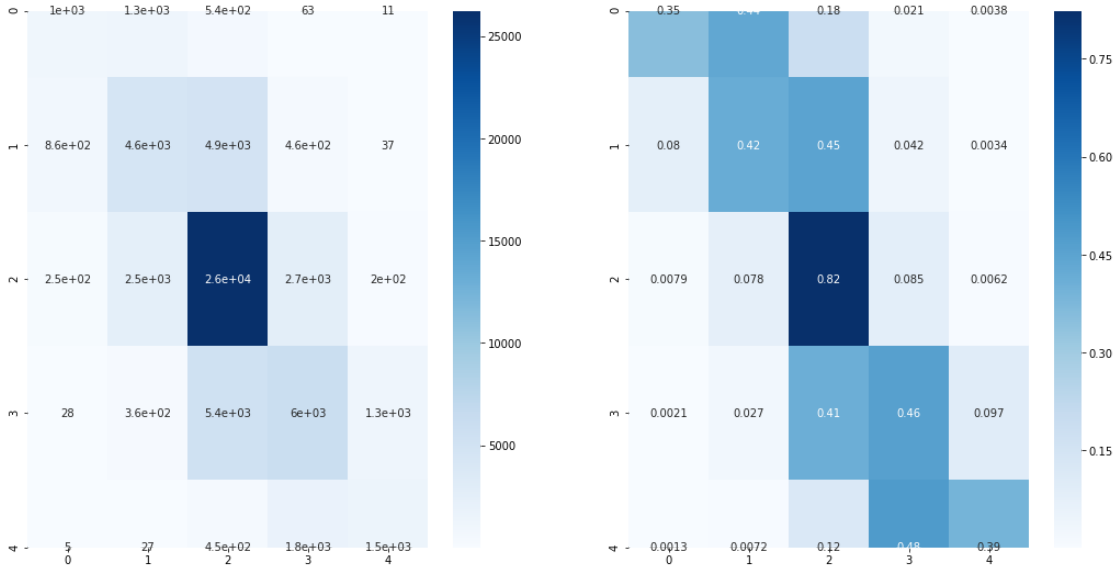


png

```
----------------------------------------
##SVM
----------------------------------------


/usr/local/lib/python3.7/site-packages/sklearn/svm/base.py:929:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  "the number of iterations.", ConvergenceWarning)


Confusion Matrices: Non-normalized and Normalized
```



png

==========

BY SENTIMENT

==========

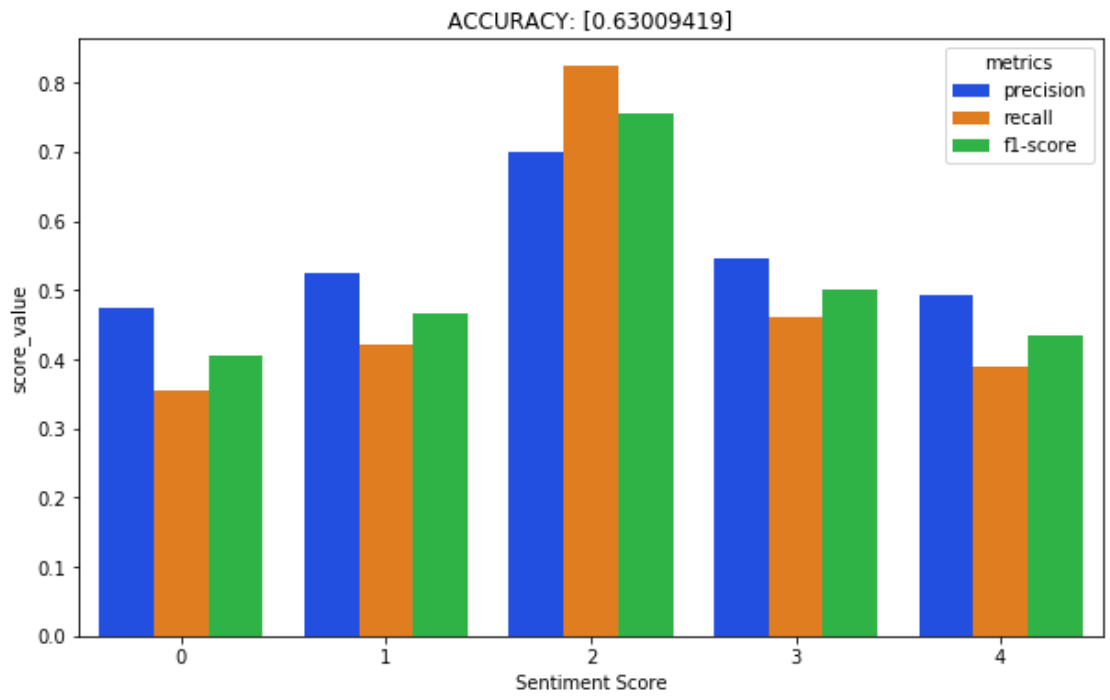
	0	1	2	3	4
precision	0.47	0.52	0.70	0.55	0.49
recall	0.35	0.42	0.82	0.46	0.39
f1-score	0.41	0.47	0.76	0.50	0.44
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

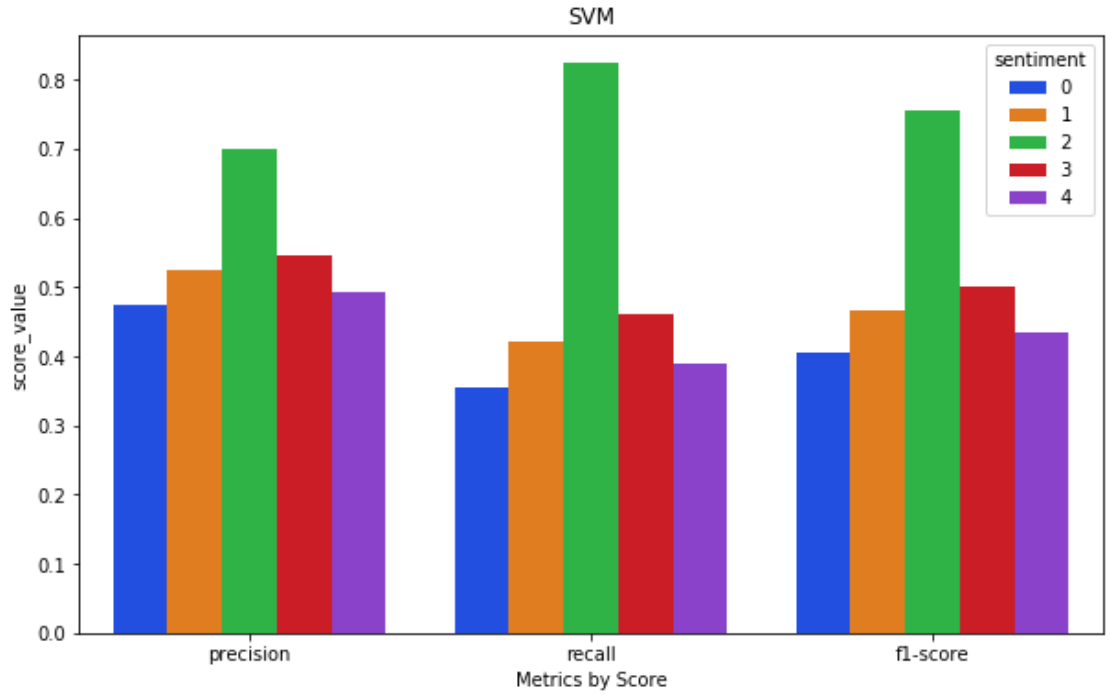
BY PERFORMANCE

==========

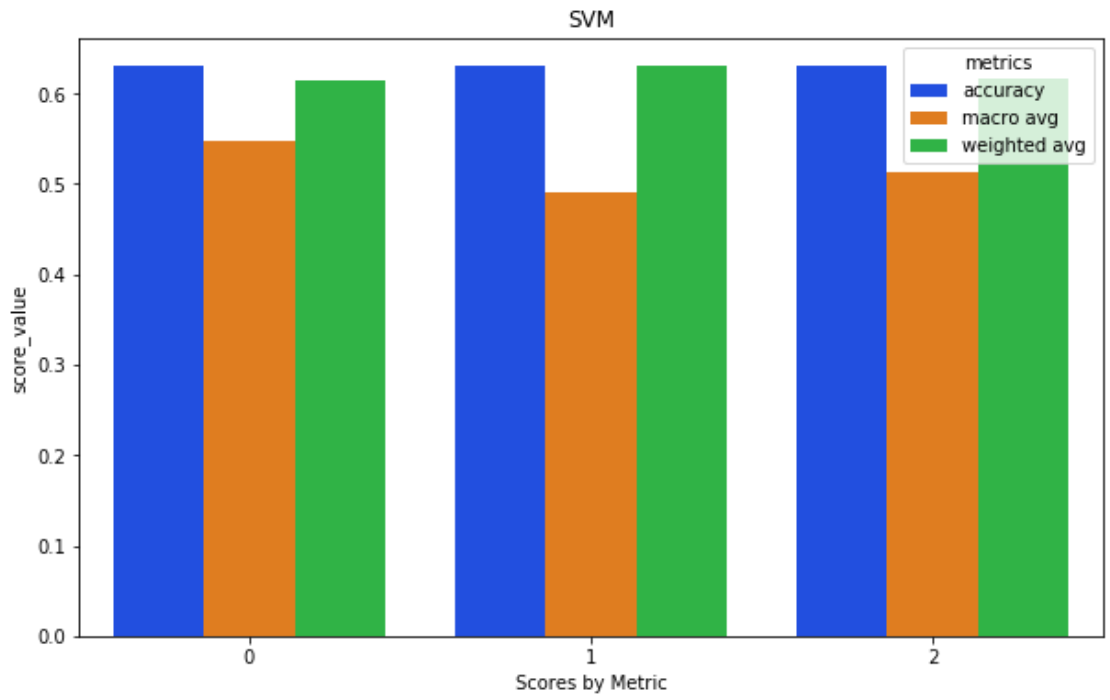
	accuracy	macro avg	weighted avg
precision	0.63	0.55	0.61
recall	0.63	0.49	0.63
f1-score	0.63	0.51	0.62
support	0.63	62424.00	62424.00


ACCURACY: [0.63009419]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

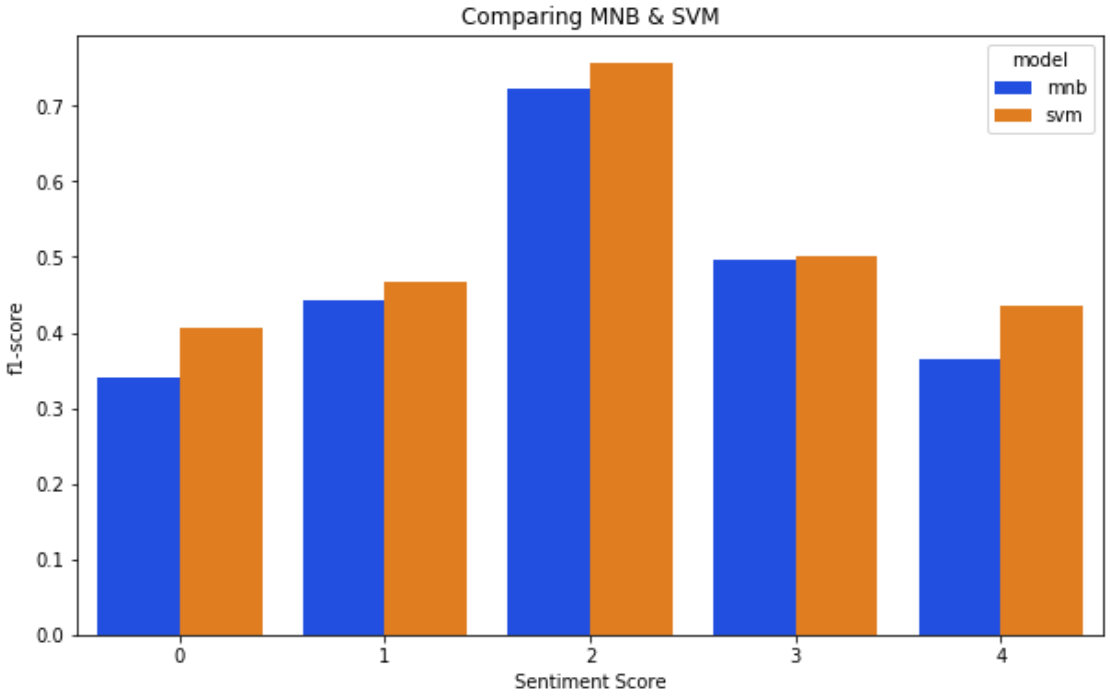
png


SVM
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

png



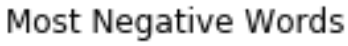
png



png

MNB FEATURES**********

```
<Figure size 432x288 with 0 Axes>
```



0	1
-6.629714522576447	time
-6.61913241324591	characters
-6.608661113378615	minutes
-6.608661113378615	story
-6.598298326343068	comedy
-6.377218759249709	just
-5.825901774128982	like
-5.663620968218473	bad
-5.5205201245778	film
-5.0096945008118094	movie

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-11.17300930484645	10 course
-11.17300930484645	10 year
-11.17300930484645	100 minute
-11.17300930484645	100 years
-11.17300930484645	101 minutes
-11.17300930484645	101 premise
-11.17300930484645	102
-11.17300930484645	102 minute
-11.17300930484645	10th
-11.17300930484645	10th grade

png

```
SVM FEATURES******************************
```

```
<Figure size 432x288 with 0 Axes>
```

Most Negative Words

0	1
1.7394374517757494	charm laughs
1.7467982397926674	unappealing
1.7584453144980825	unwatchable
1.7990447702053058	unbearable
1.8031737556021281	waste
1.8061703791719452	utterly incompetent
1.857408892105231	disgusting
1.9182461905317276	distasteful
1.9598715575422538	pompous
1.9628019012320392	garbage

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-2.01254345708511	good good
-1.9906388225385072	director ca
-1.82457847167496	variation
-1.7319158724061963	bad cinema
-1.597550356784629	acting ensemble
-1.5690014001368477	swimfan
-1.4259737117653501	payoff audience
-1.4141037716451264	luridly
-1.3834234805439847	wannabe comedy
-1.369918482440678	took

png

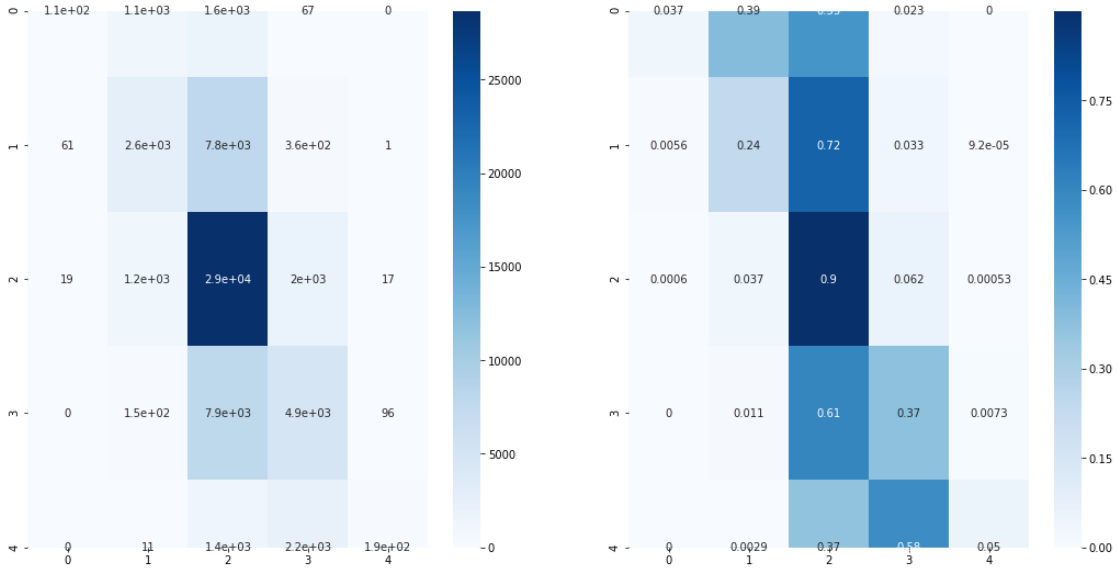
```
Vectorizer Settings for 3_TfidfVectorizer
```

```
<Figure size 432x288 with 0 Axes>
```

index	0
analyzer	word
binary	False
decode_error	strict
dtype	<class 'numpy.float64'>
encoding	latin-1
input	content
lowercase	True
max_df	1.0
max_features	
min_df	5
ngram_range	(1, 1)
norm	l2
preprocessor	
smooth_idf	True
stop_words	english
strip_accents	
sublinear_tf	False
token_pattern	(?u)\b\w\w+\b
tokenizer	
use_idf	True
vocabulary	

png

```
----------------------------------------
##MNB
----------------------------------------
Confusion Matrices: Non-normalized and Normalized
```



png

==========

BY SENTIMENT

==========

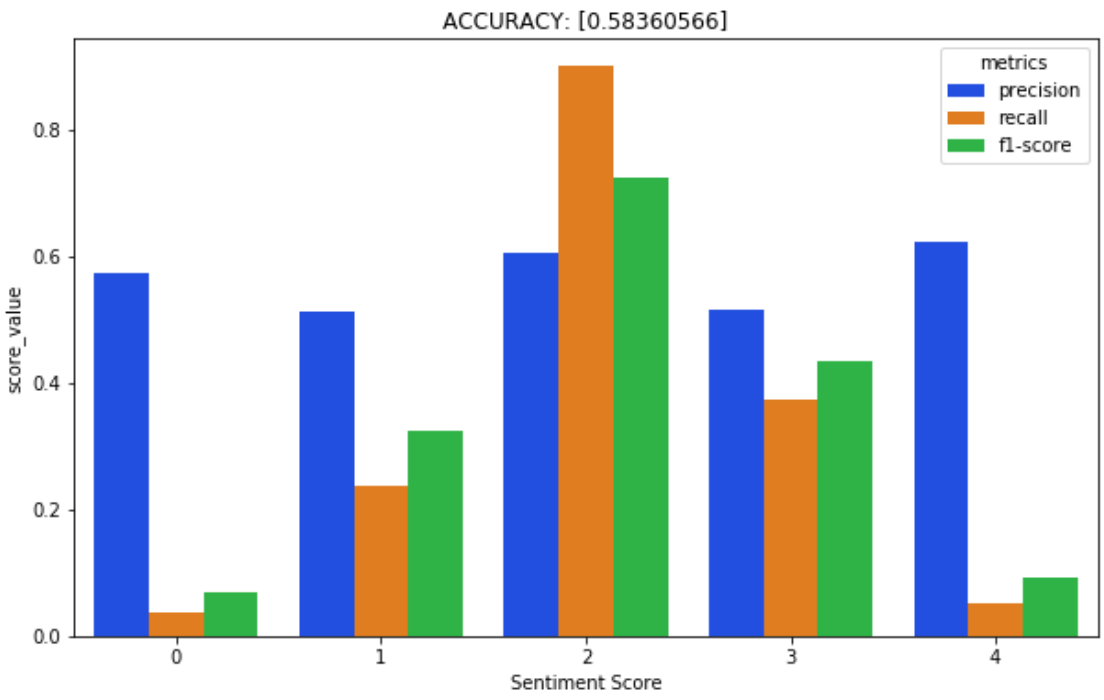
	0	1	2	3	4
precision	0.57	0.51	0.60	0.52	0.62
recall	0.04	0.24	0.90	0.37	0.05
f1-score	0.07	0.33	0.72	0.43	0.09
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

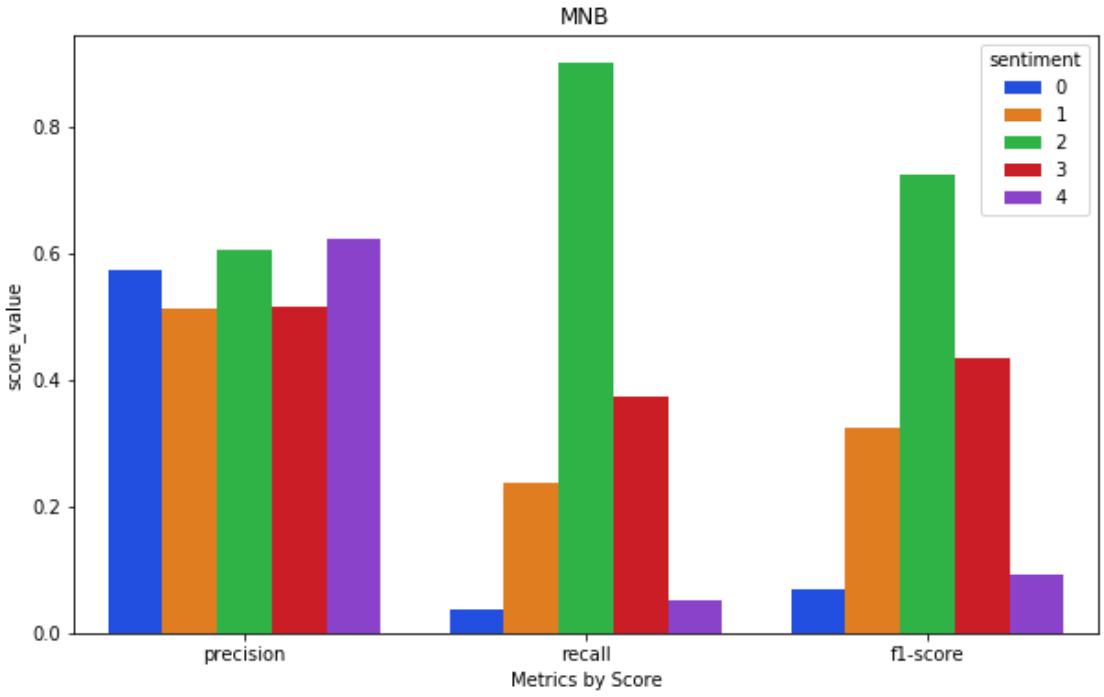
BY PERFORMANCE

==========

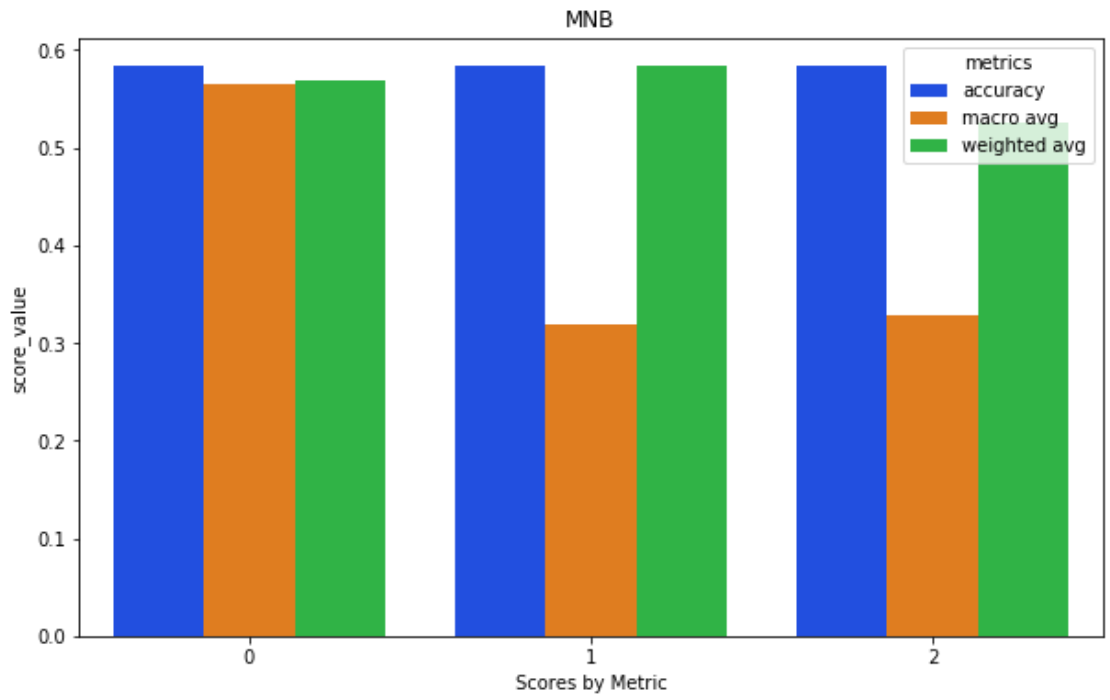
	accuracy	macro avg	weighted avg
precision	0.58	0.57	0.57
recall	0.58	0.32	0.58
f1-score	0.58	0.33	0.53
support	0.58	62424.00	62424.00


ACCURACY: [0.58360566]
metrics
precision
recall
f1-score
score_value
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
Sentiment Score

png


MNB
sentiment
0
1
2
3
4
score_value
0.0
0.2
0.4
0.6
0.8
precision
recall
f1-score
Metrics by Score

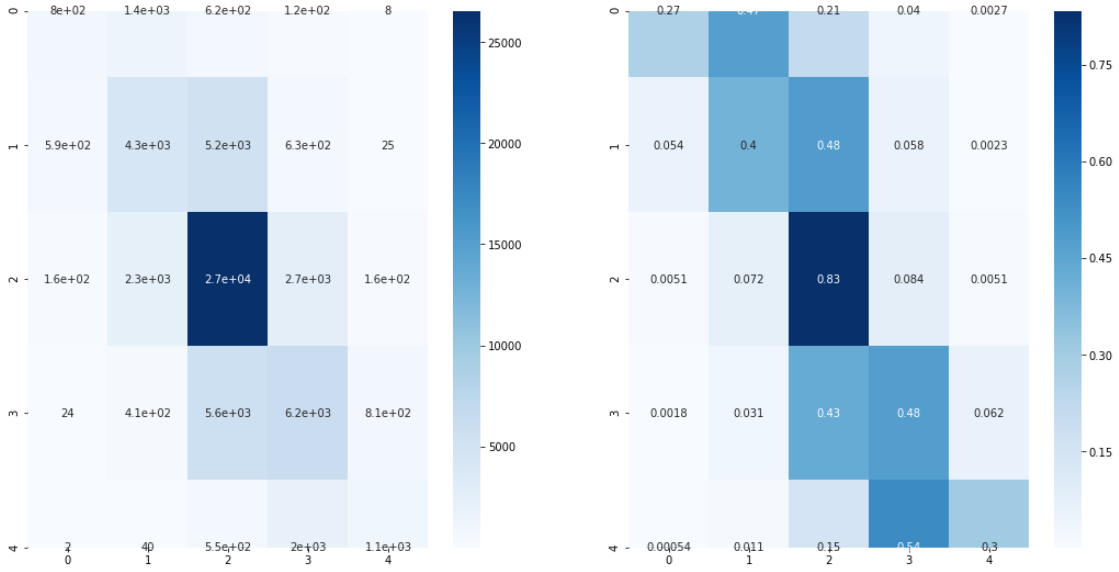
png



png

##SVM

Confusion Matrices: Non-normalized and Normalized



png

==========

BY SENTIMENT

==========

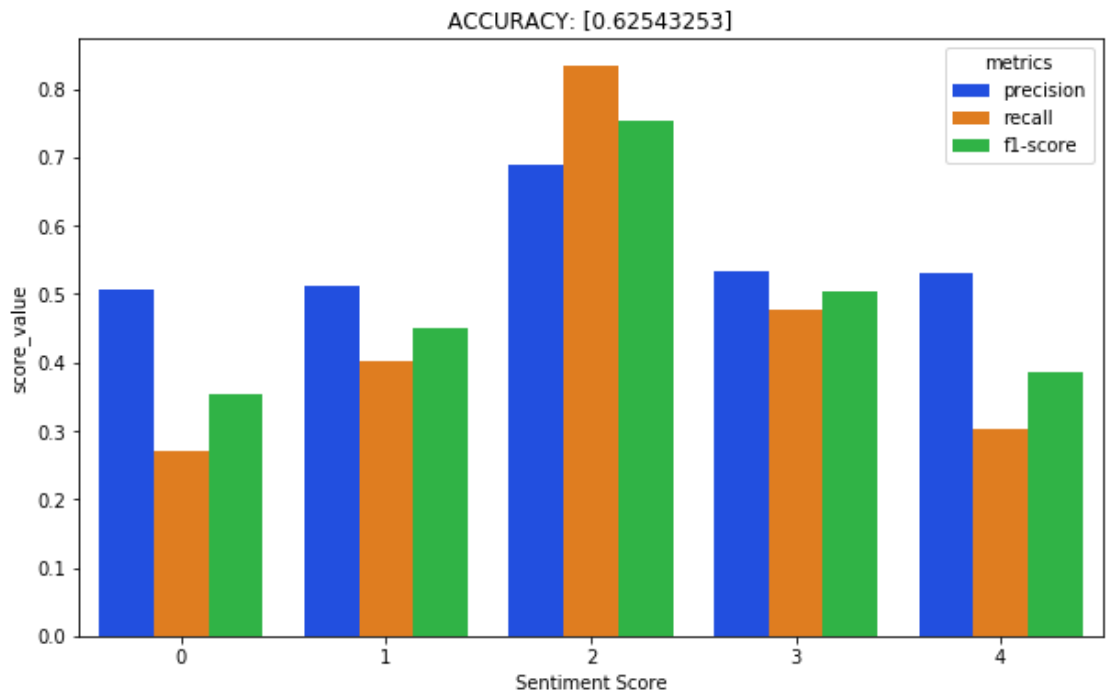
	0	1	2	3	4
precision	0.51	0.51	0.69	0.53	0.53
recall	0.27	0.40	0.83	0.48	0.30
f1-score	0.35	0.45	0.75	0.50	0.39
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

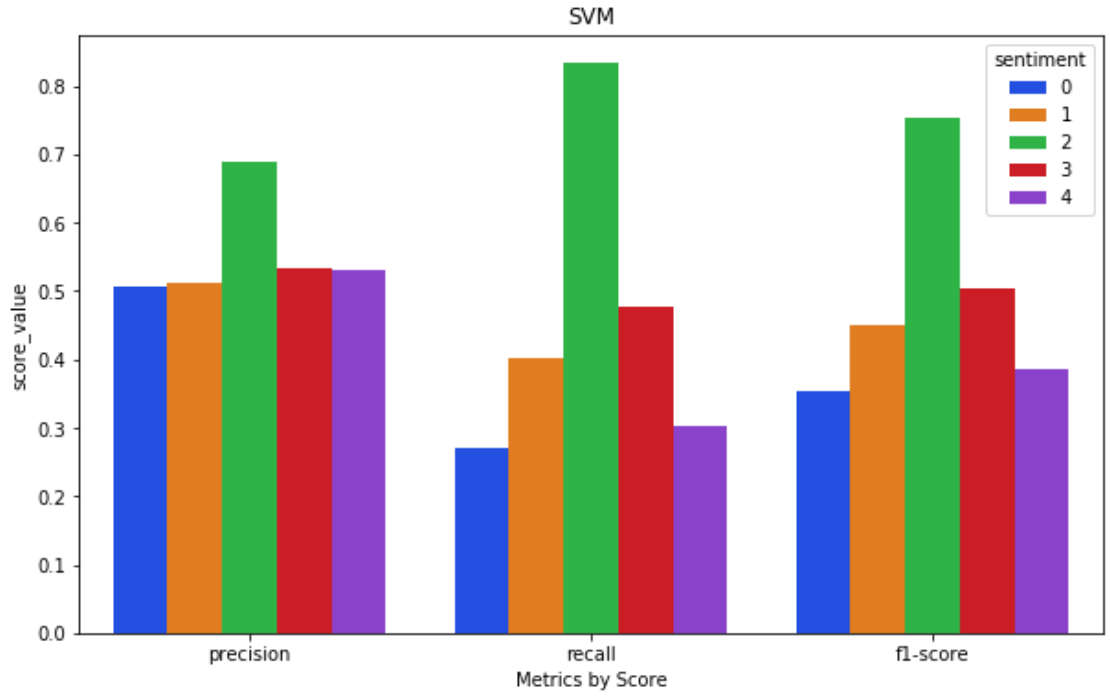
BY PERFORMANCE

==========

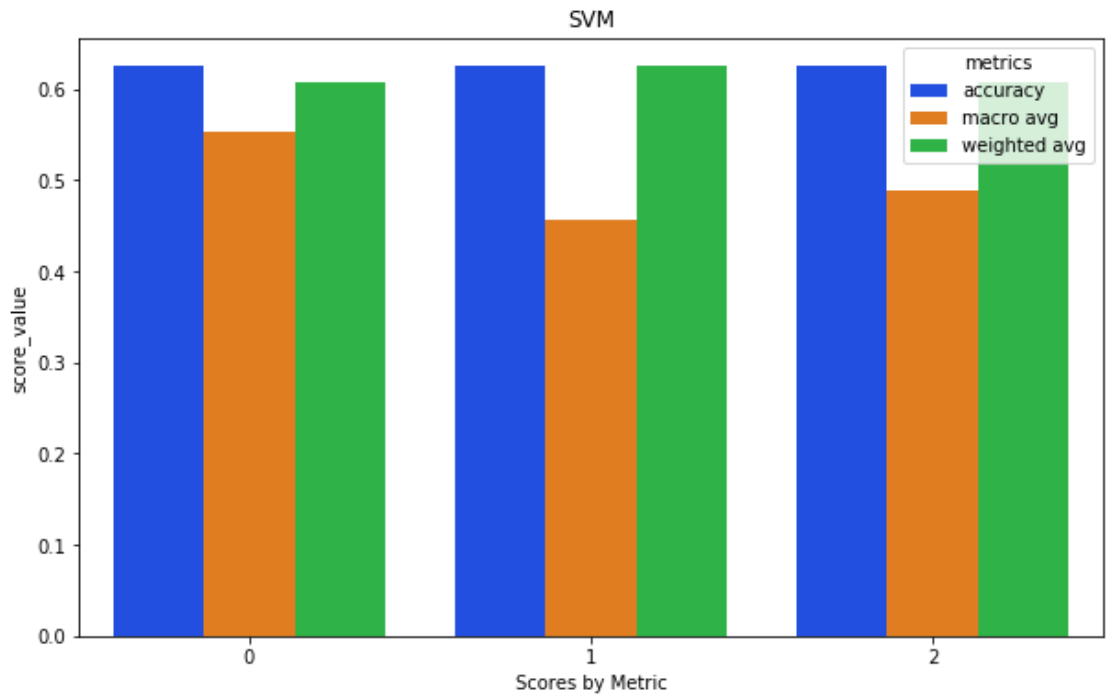
	accuracy	macro avg	weighted avg
precision	0.63	0.55	0.61
recall	0.63	0.46	0.63
f1-score	0.63	0.49	0.61
support	0.63	62424.00	62424.00


ACCURACY: [0.62543253]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

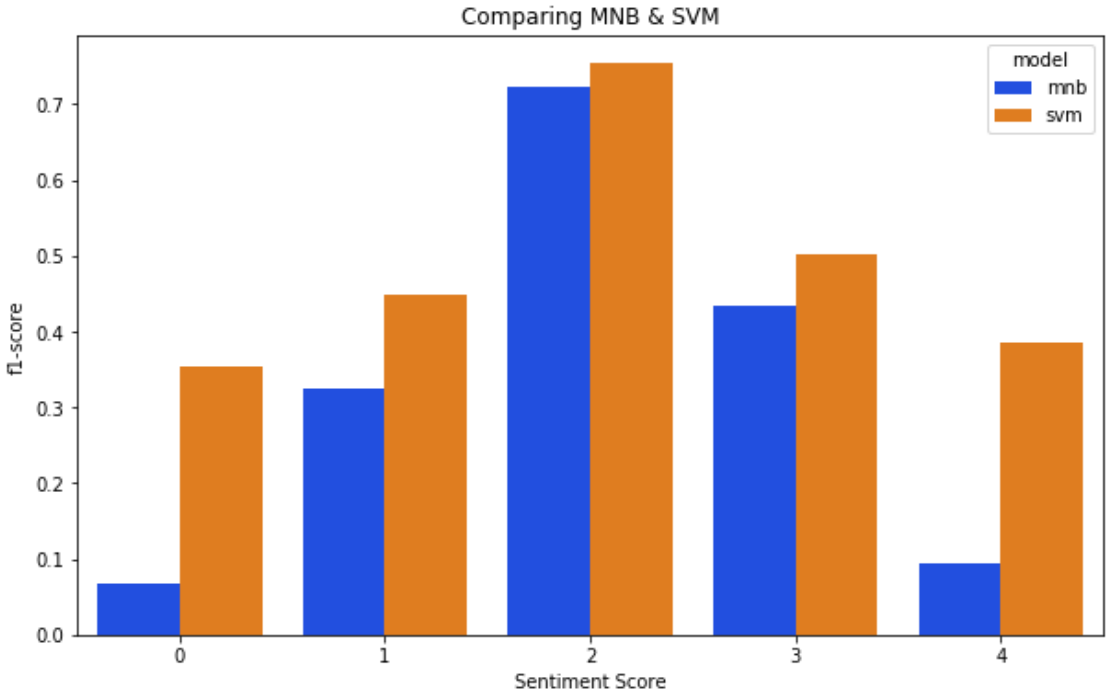
png


SVM
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

png



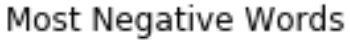
png



png

MNB FEATURES**********

```
<Figure size 432x288 with 0 Axes>
```



0	1
-6.645979011120781	time
-6.62020923362483	does
-6.60369986228579	minutes
-6.517311233308192	dull
-6.355845020937053	just
-6.131355008437069	worst
-6.029810232070883	like
-5.792621365210278	film
-5.411480448479903	bad
-5.186697744581525	movie

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-9.958278961750986	102
-9.958278961750986	10th
-9.958278961750986	127
-9.958278961750986	13th
-9.958278961750986	14
-9.958278961750986	16
-9.958278961750986	163
-9.958278961750986	168
-9.958278961750986	170
-9.958278961750986	1790

png

SVM FEATURES******************************

<Figure size 432x288 with 0 Axes>

Most Negative Words

0	1
2.0987432505975527	stinks
2.1727383182586975	worthless
2.179842026367953	worst
2.1800484032803364	distasteful
2.2009545894592875	unwatchable
2.2123829092678204	unbearable
2.2381374771933076	meaningless
2.4621411041306542	stinker
2.544275163254315	disappointment
2.563481596126207	disgusting

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-1.5438542784214793	giddy
-1.3211650816819376	collar
-1.145358610054194	victim
-1.1388809661965995	activity
-1.0972856903250492	innocence
-1.0915543900190494	loving
-1.0786327265966278	rehashes
-1.0635187810310303	engrossing
-1.047372932271783	modern
-1.0303688394248303	morlocks

png

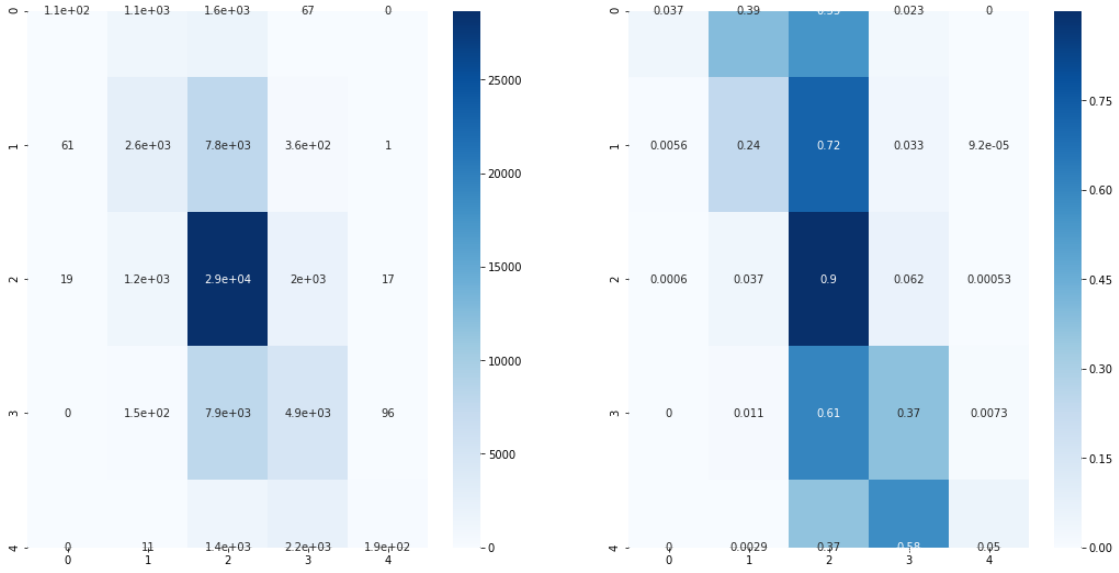
```
Vectorizer Settings for 4_TfidfVectorizer
```

```
<Figure size 432x288 with 0 Axes>
```

index	0
analyzer	word
binary	False
decode_error	strict
dtype	<class 'numpy.float64'>
encoding	latin-1
input	content
lowercase	True
max_df	0.5
max_features	
min_df	5
ngram_range	(1, 1)
norm	l2
preprocessor	
smooth_idf	True
stop_words	english
strip_accents	
sublinear_tf	False
token_pattern	(?u)\b\w\w+\b
tokenizer	
use_idf	True
vocabulary	

png

```
----------------------------------------
##MNB
----------------------------------------
Confusion Matrices: Non-normalized and Normalized
```



png

==========

BY SENTIMENT

==========

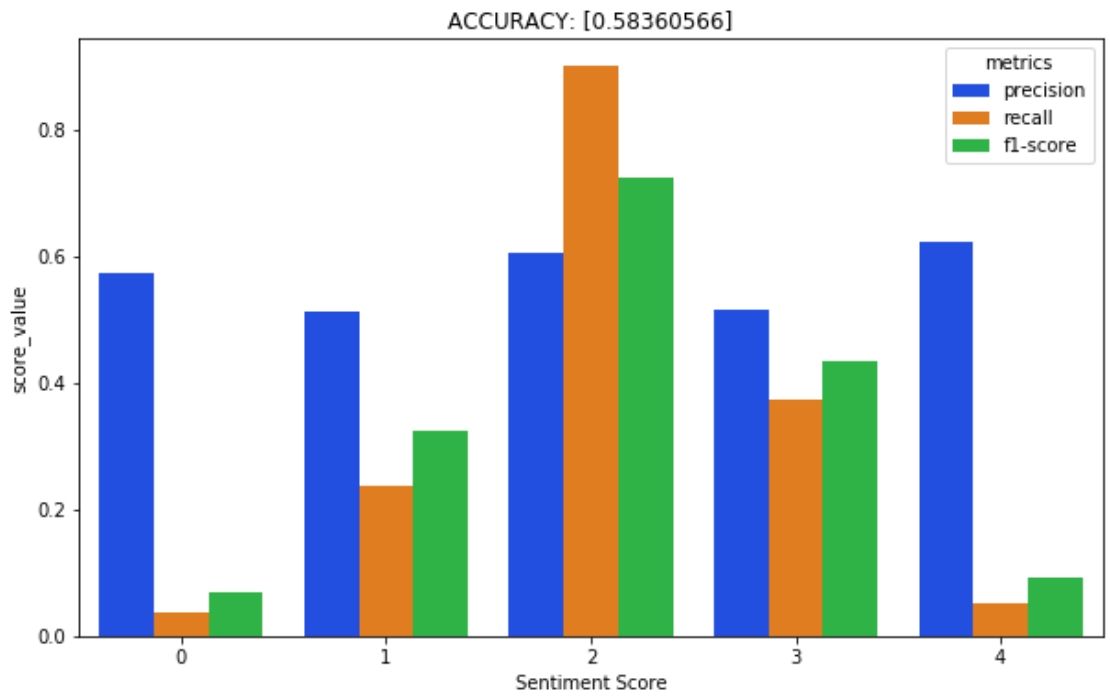
	0	1	2	3	4
precision	0.57	0.51	0.60	0.52	0.62
recall	0.04	0.24	0.90	0.37	0.05
f1-score	0.07	0.33	0.72	0.43	0.09
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

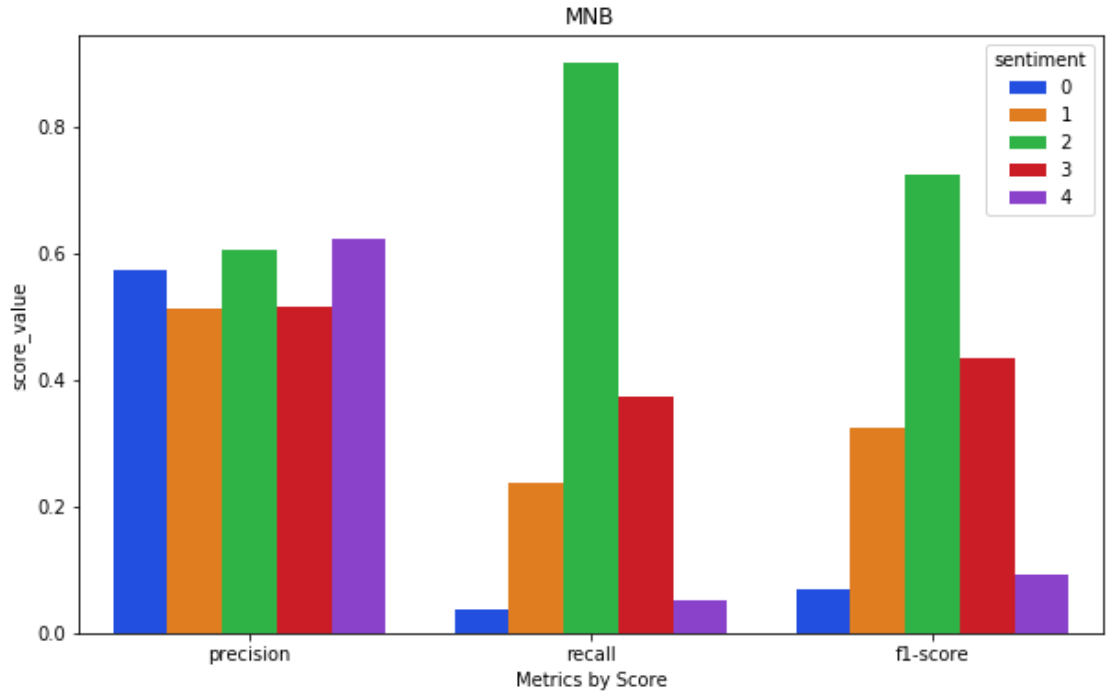
BY PERFORMANCE

==========

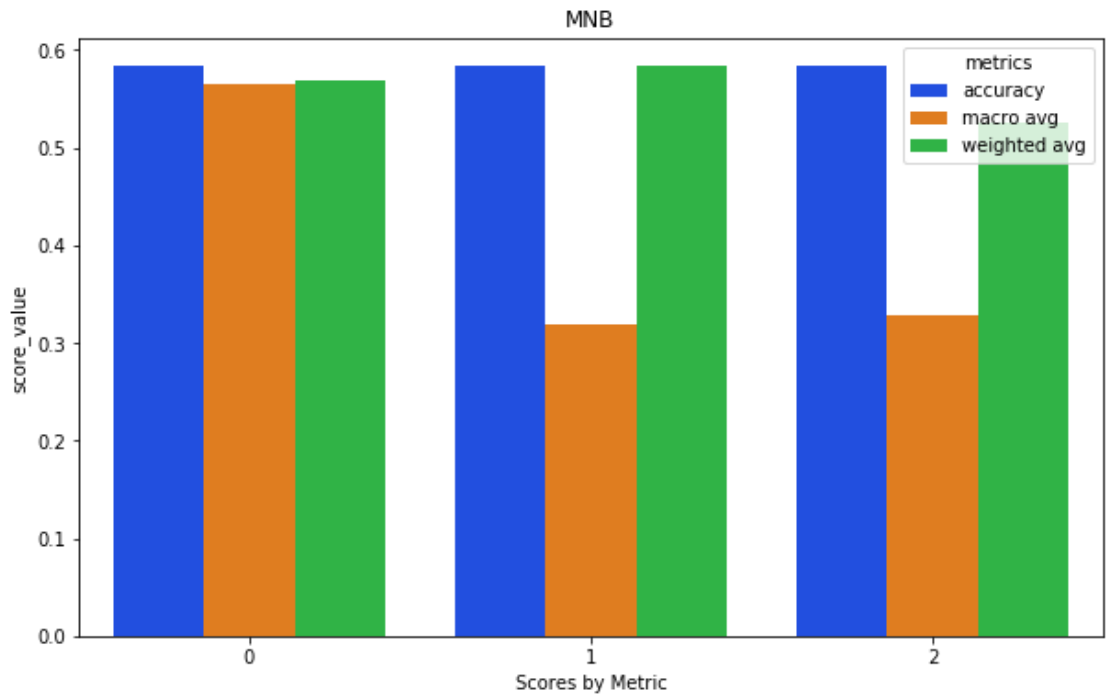
	accuracy	macro avg	weighted avg
precision	0.58	0.57	0.57
recall	0.58	0.32	0.58
f1-score	0.58	0.33	0.53
support	0.58	62424.00	62424.00


ACCURACY: [0.58360566]
metrics
precision
recall
f1-score
score_value
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
Sentiment Score

png


MNB
sentiment
0
1
2
3
4
score_value
0.0
0.2
0.4
0.6
0.8
precision
recall
f1-score
Metrics by Score

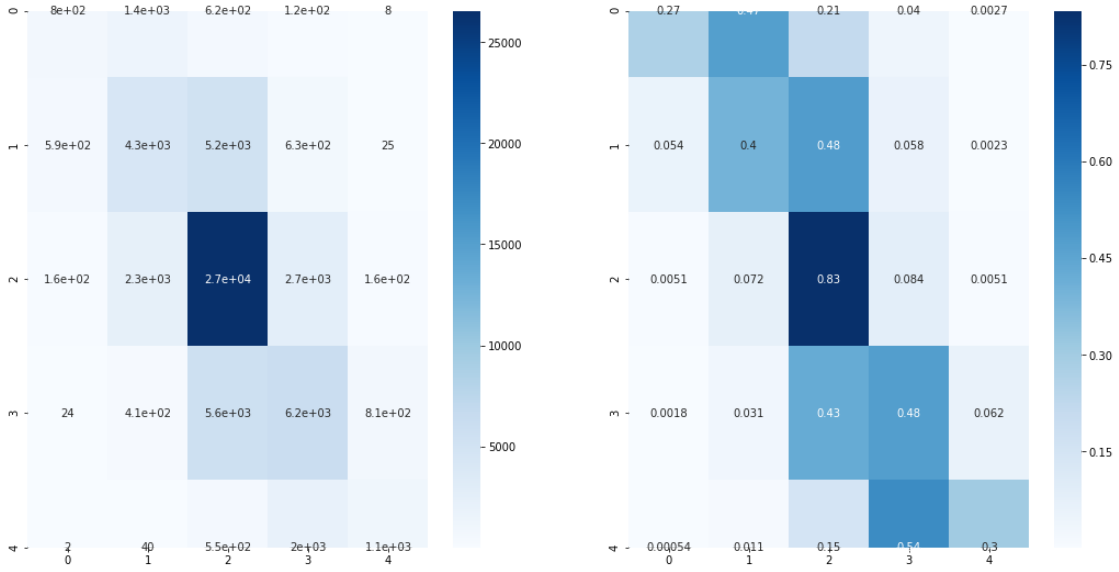
png



png

##SVM

Confusion Matrices: Non-normalized and Normalized



png

==========

BY SENTIMENT

==========

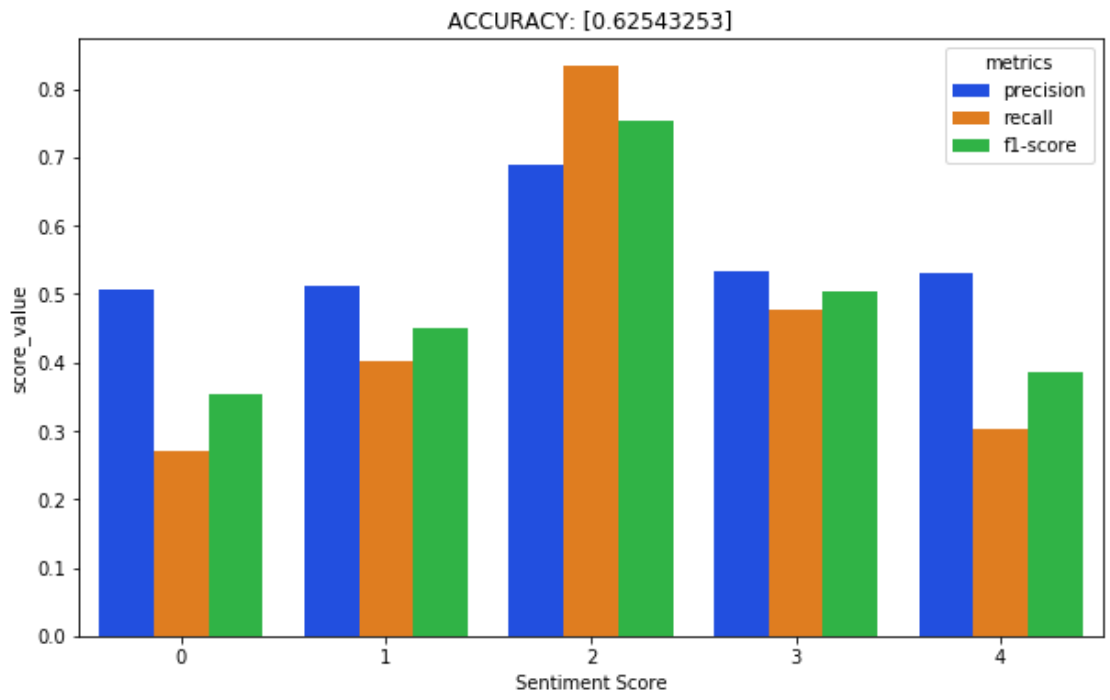
	0	1	2	3	4
precision	0.51	0.51	0.69	0.53	0.53
recall	0.27	0.40	0.83	0.48	0.30
f1-score	0.35	0.45	0.75	0.50	0.39
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

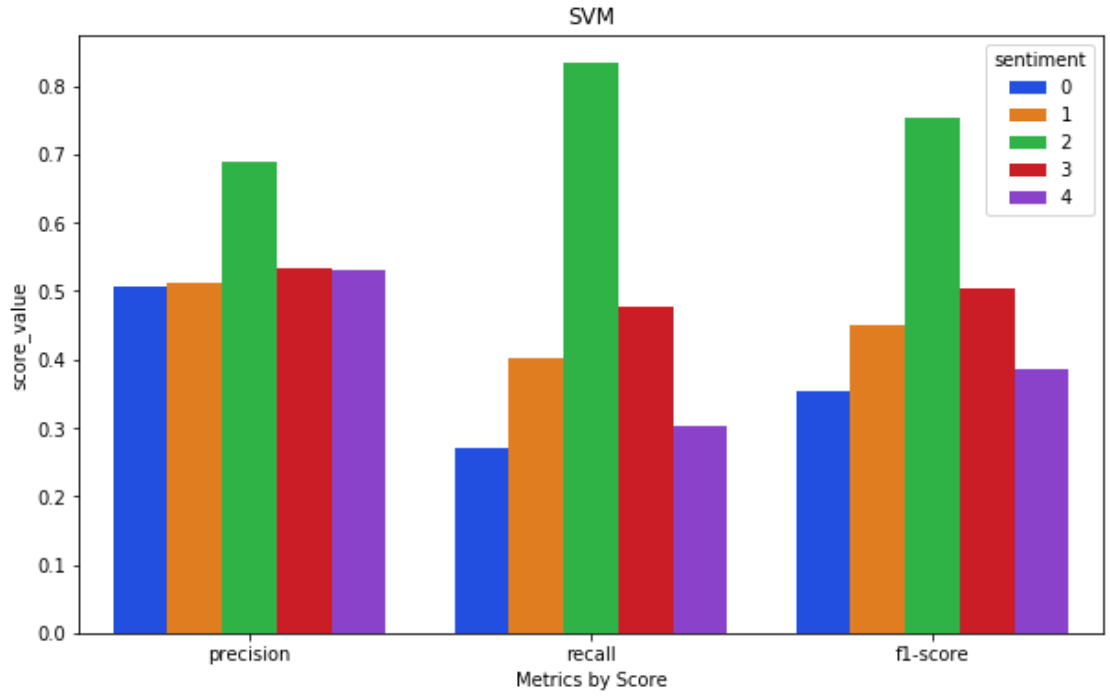
BY PERFORMANCE

==========

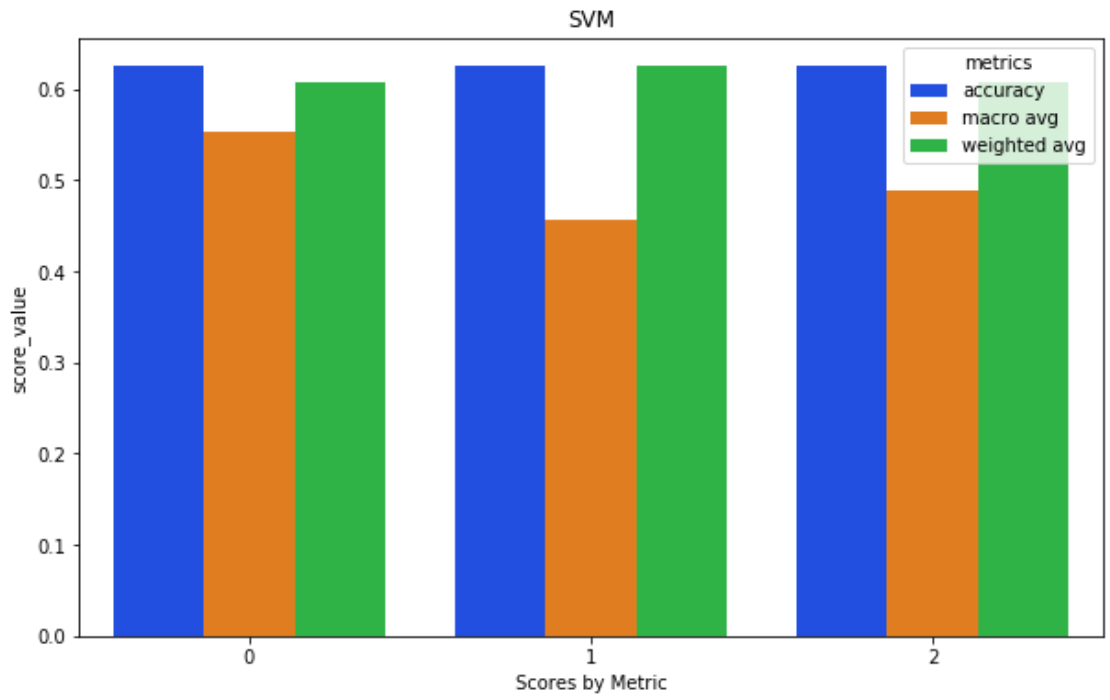
	accuracy	macro avg	weighted avg
precision	0.63	0.55	0.61
recall	0.63	0.46	0.63
f1-score	0.63	0.49	0.61
support	0.63	62424.00	62424.00


ACCURACY: [0.62543253]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

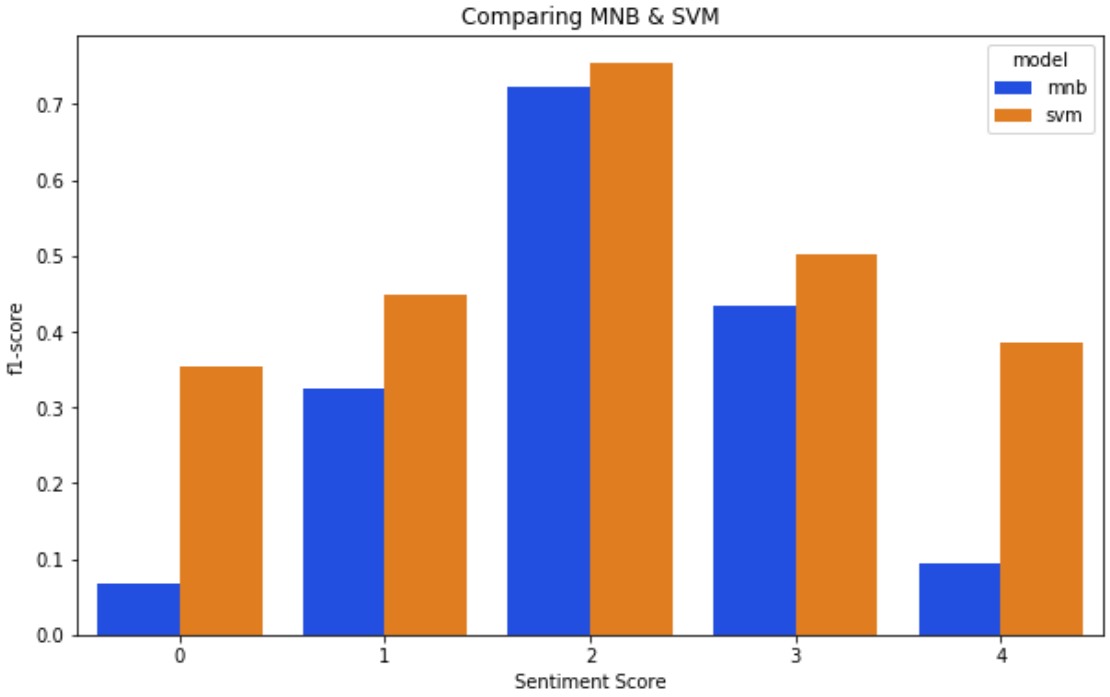
png


SVM
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

png



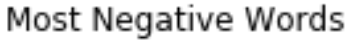
png



png

MNB FEATURES**********

```
<Figure size 432x288 with 0 Axes>
```



0	1
-6.645979011120781	time
-6.62020923362483	does
-6.60369986228579	minutes
-6.517311233308192	dull
-6.355845020937053	just
-6.131355008437069	worst
-6.029810232070883	like
-5.792621365210278	film
-5.411480448479903	bad
-5.186697744581525	movie

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-9.958278961750986	102
-9.958278961750986	10th
-9.958278961750986	127
-9.958278961750986	13th
-9.958278961750986	14
-9.958278961750986	16
-9.958278961750986	163
-9.958278961750986	168
-9.958278961750986	170
-9.958278961750986	1790

png

SVM FEATURES******************************

<Figure size 432x288 with 0 Axes>

Most Negative Words

0	1
2.0987434806303575	stinks
2.1727399209560057	worthless
2.1798402339086156	worst
2.180051728022837	distasteful
2.2009534302593403	unwatchable
2.2123845079612052	unbearable
2.238137890471789	meaningless
2.462144180117136	stinker
2.544277015151644	disappointment
2.5634861159698805	disgusting

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-1.5438546289230826	giddy
-1.32116264072656	collar
-1.145359635087421	victim
-1.138882236213439	activity
-1.0972860068399157	innocence
-1.0915549939243296	loving
-1.0786284021579768	rehashes
-1.0635211234835584	engrossing
-1.0473689102093944	modern
-1.0303639090162104	morlocks

png

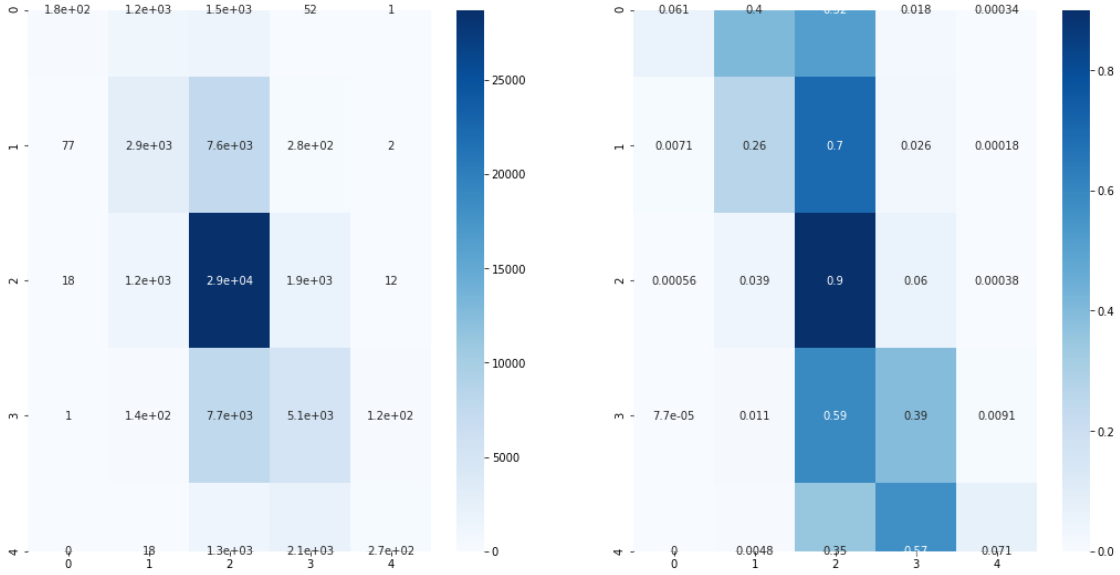
```
Vectorizer Settings for 5_TfidfVectorizer
```

```
<Figure size 432x288 with 0 Axes>
```

index	0
analyzer	word
binary	False
decode_error	strict
dtype	<class 'numpy.float64'>
encoding	latin-1
input	content
lowercase	True
max_df	1.0
max_features	
min_df	5
ngram_range	(1, 2)
norm	l2
preprocessor	
smooth_idf	True
stop_words	english
strip_accents	
sublinear_tf	False
token_pattern	(?u)\b\w\w+\b
tokenizer	
use_idf	True
vocabulary	

png

```
----------------------------------------
##MNB
----------------------------------------
Confusion Matrices: Non-normalized and Normalized
```



png

==========

BY SENTIMENT

==========

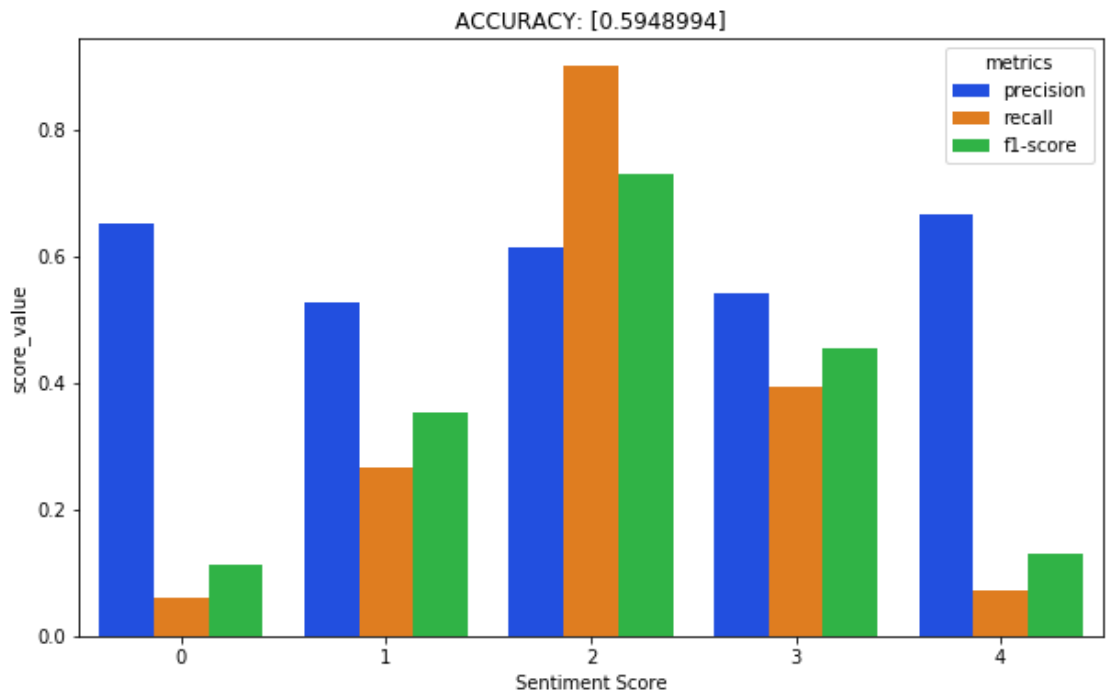
	0	1	2	3	4
precision	0.65	0.53	0.61	0.54	0.67
recall	0.06	0.26	0.90	0.39	0.07
f1-score	0.11	0.35	0.73	0.45	0.13
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

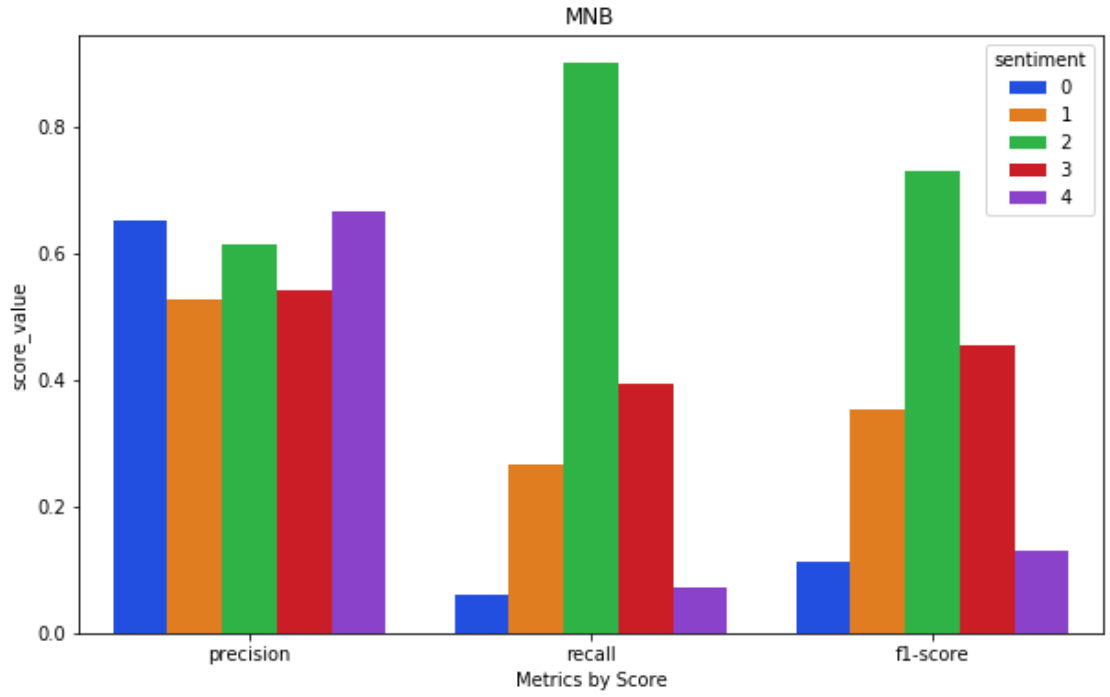
BY PERFORMANCE

==========

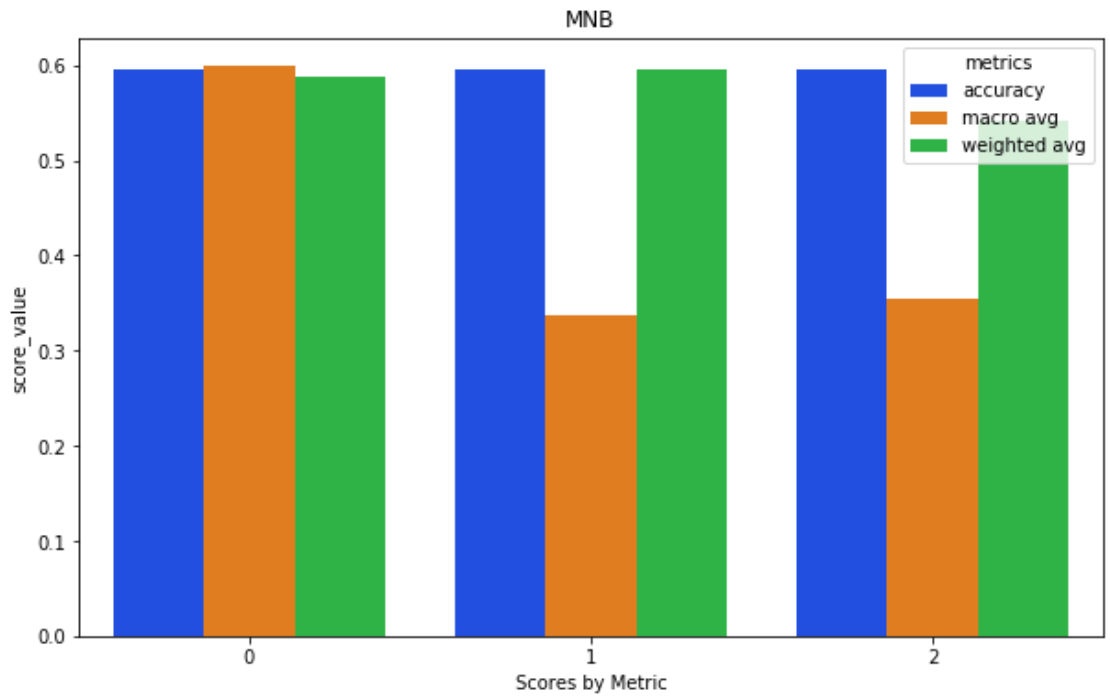
	accuracy	macro avg	weighted avg
precision	0.59	0.60	0.59
recall	0.59	0.34	0.59
f1-score	0.59	0.36	0.54
support	0.59	62424.00	62424.00


ACCURACY: [0.5948994]
metrics
precision
recall
f1-score
score_value
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
Sentiment Score

png


MNB
sentiment
0
1
2
3
4
score_value
0.0
0.2
0.4
0.6
0.8
precision
recall
f1-score
Metrics by Score

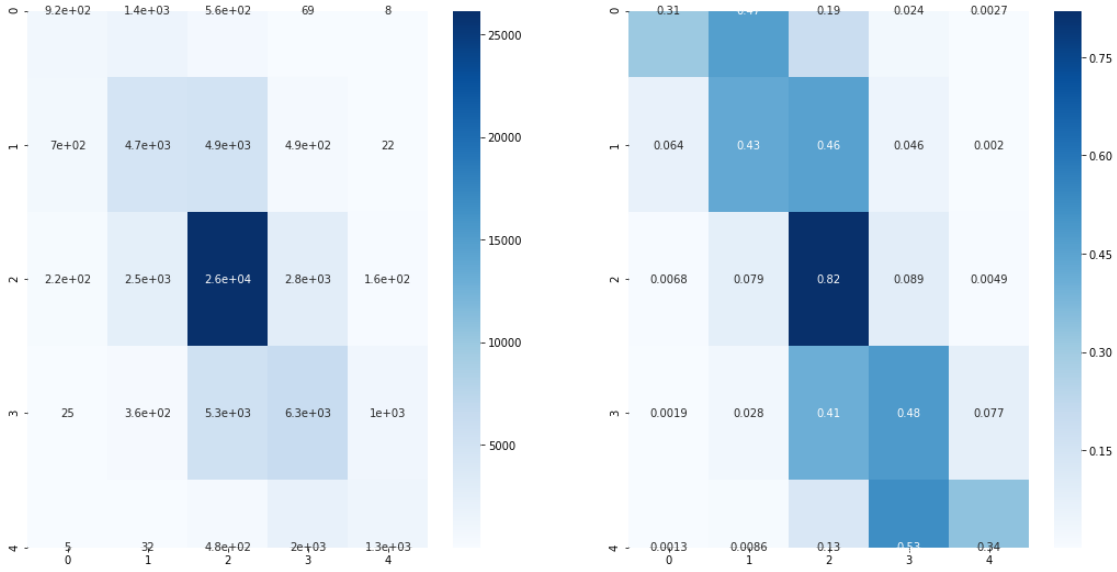
png



png

##SVM

Confusion Matrices: Non-normalized and Normalized



png

==========

BY SENTIMENT

==========

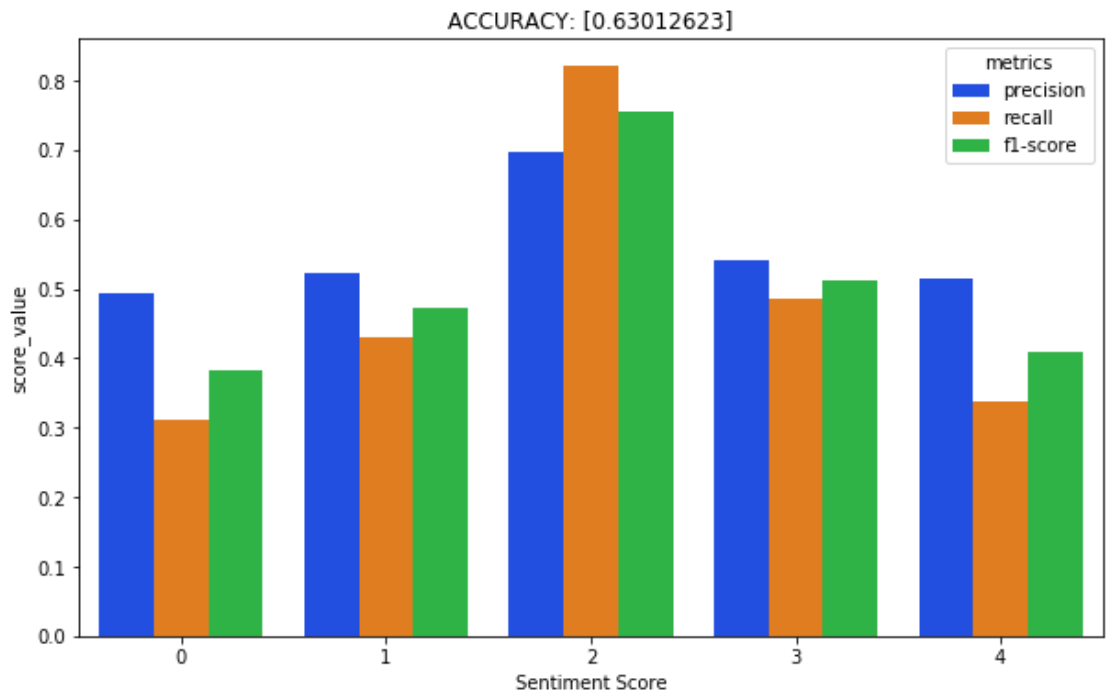
	0	1	2	3	4
precision	0.49	0.52	0.70	0.54	0.52
recall	0.31	0.43	0.82	0.48	0.34
f1-score	0.38	0.47	0.75	0.51	0.41
support	2931.00	10824.00	31864.00	13068.00	3737.00

==========

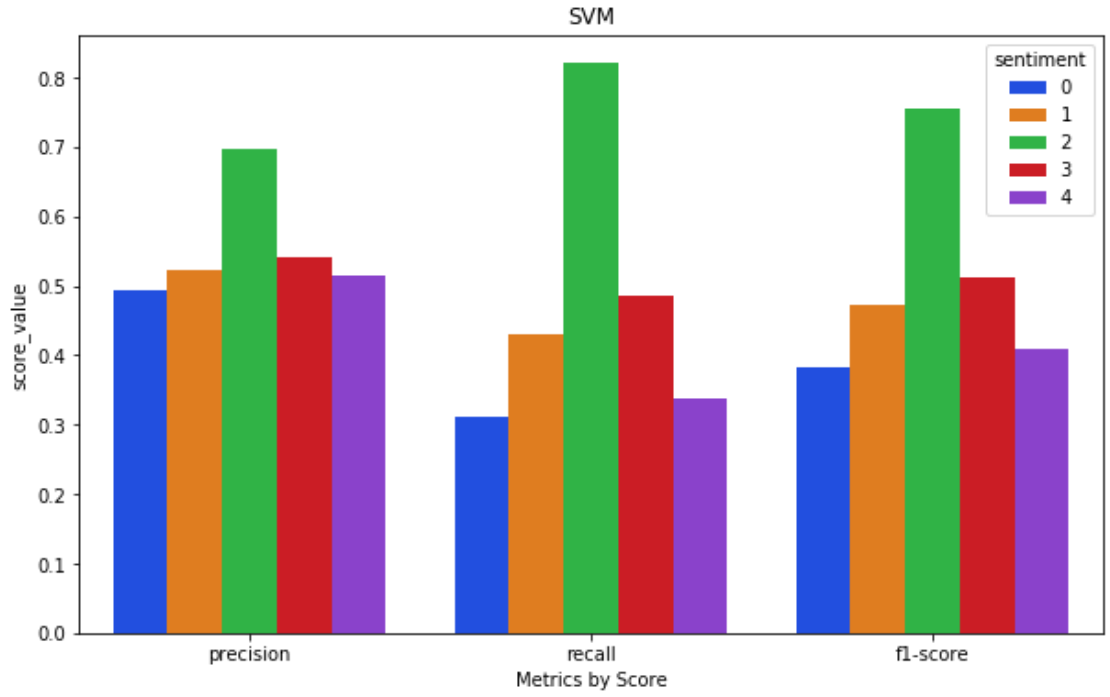
BY PERFORMANCE

==========

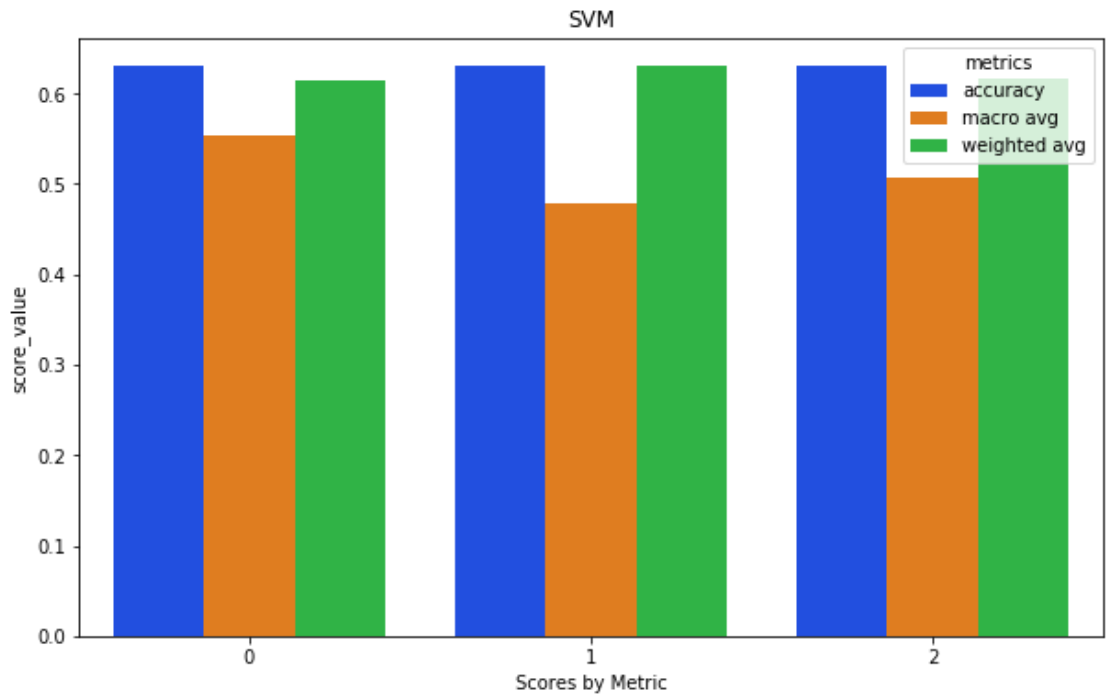
	accuracy	macro avg	weighted avg
precision	0.63	0.55	0.61
recall	0.63	0.48	0.63
f1-score	0.63	0.51	0.62
support	0.63	62424.00	62424.00


ACCURACY: [0.63012623]
metrics
precision
recall
f1-score
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
Sentiment Score

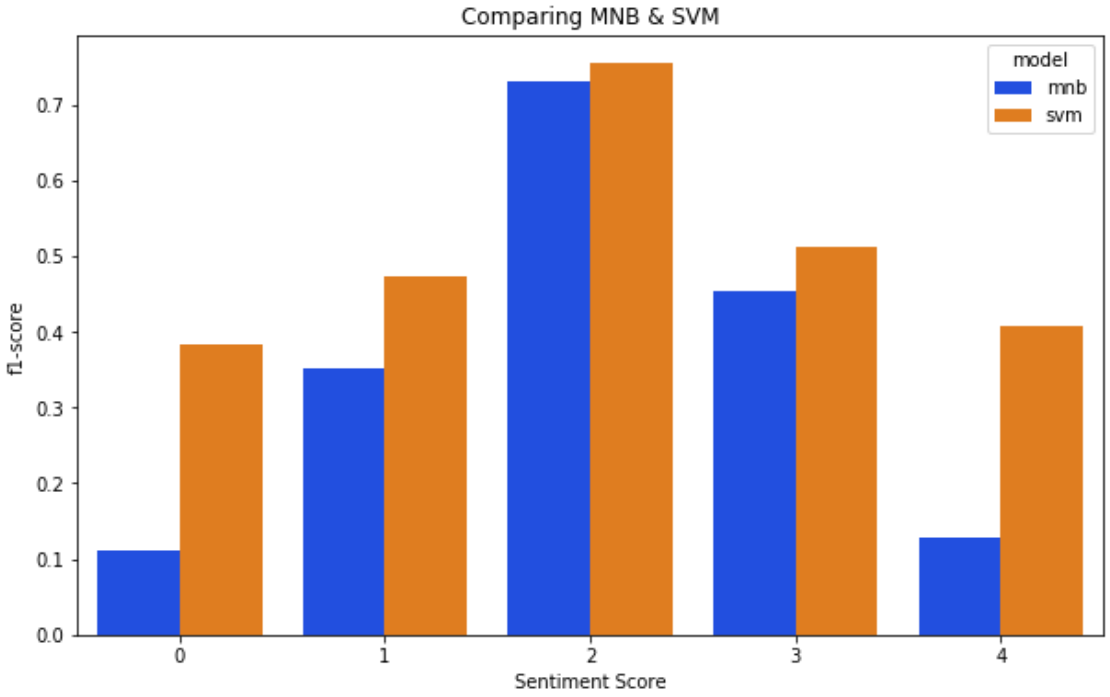
png


SVM
sentiment
0
1
2
3
4
score_value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
precision
recall
f1-score
Metrics by Score

png



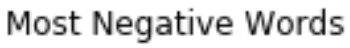
png



png

MNB FEATURES**********

```
<Figure size 432x288 with 0 Axes>
```



0	1
-7.670900924532225	story
-7.654743793752857	stupid
-7.624189077492526	mess
-7.498580938421712	dull
-7.393303232254995	just
-7.209695440732919	worst
-7.108870852486083	like
-6.852838351434956	film
-6.464253994333886	bad
-6.241813501470778	movie

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-10.730617771376618	10 course
-10.730617771376618	10 year
-10.730617771376618	100 minute
-10.730617771376618	100 years
-10.730617771376618	101 minutes
-10.730617771376618	101 premise
-10.730617771376618	102
-10.730617771376618	102 minute
-10.730617771376618	10th
-10.730617771376618	10th grade

png

SVM FEATURES******************************

<Figure size 432x288 with 0 Axes>

Most Negative Words

0	1
2.0887644406961727	awful
2.1120425608294697	unwatchable
2.1290846143054756	unbearable
2.1416313582203275	entirely witless
2.167454286574856	distasteful
2.2745522113494467	disgusting
2.310963825656959	garbage
2.3268852288018356	charm laughs
2.442728747606491	waste
2.6823976594637546	disappointment

png

```
<Figure size 432x288 with 0 Axes>
```

Most Positive Words

0	1
-1.3371952463976153	good good
-1.2940370416251212	variation
-1.09761277171423	just like
-1.0783420554955498	going really
-1.0729994720194633	man garbage
-1.0684304547335028	lightness
-1.0646209240922557	awful lot
-1.0362481443785327	loving
-1.0270172512119873	appear
-0.9976743606124305	movie way

ALL TOGETHER NOW

TASK 3 `

RESULTS B

TASK 1

TEST 1:

Ran both MNB and SVM with Vectorizer 1. MNB came back with a lot of numbers, so for Vectorizer Test 2, I edited the “token_pattern” parameters to explicitly include only alpha words .

```
token_pattern=r'(?u)\b[a-zA-Z]{2,}\b'
```

TEST 3:

This tested a non-binary unigram CountVectorizer vectorizer with the same token pattern as Test 2. Results were nearly identical as Test 2, interestly SVM performed slightly less well.

TEST 4:

MOTIVATION: To get a better accuracy

MANIPULATING: grams -- trying bi-grams this time.

Overall, our best accuracy yet -- SVM removing bigrams yielded 63%. Interestingly, bigrams with MNB decreased the accuracy.

TEST 5:

MOTIVATION: To get a better accuracy

MANIPULATING: grams -- trying bi-grams this time WITH new tokenizer

New tokenizer increased the performance of both SVM and MNB by a fraction of a percent.

TEST 6:

MOTIVATION: To get a better accuracy

MANIPULATING: Vectorizer function -- switching from CountVectorizer to TfidfVectorizer

SVM with unigrams performed one percentage point better with CV than with TFIDF.

.

TEST 7:

MOTIVATION: To get a better accuracy

MANIPULATING: Vectorizer function -- switching from CountVectorizer to TfidfVectorizer, now with the new tokenizer.

Interestingly, removing numbers ('new tokenizer') didn't make much of a difference with TFIDF.

TEST 8:

MOTIVATION: To get a better accuracy

MANIPULATING: Vectorizer function -- switching from CountVectorizer to TfidfVectorizer, now with bigrams

SVM Test 5 is still best performer. So Bigrams + CV + SVM is current best performer.

.

TEST 9:

MOTIVATION: To get a better accuracy

MANIPULATING: Vectorizer function -- switching from CountVectorizer to TfidfVectorizer, now with bigrams, now with the new tokenizer

MNB stayed the same as Test 8 (new tokenizer with bigrams didn't impact) but svm increased by 1/10th of a percent from test 8.

TASK 3

KAGGLE 1 (trained on 60%) 0.58792

KAGGLE 2 (trained on whole) 0.59145

KAGGLE 3 (with my best) 0.60494

KAGGLE 3 (with CV best) 0.60149

Results, pt2.

The researchers realized that their own hubris was yet again to blame in this futile endeavor. Did the researchers really think that they'd be able to come up with something new and innovative to a 5 year old problem? Did the researchers really think that a budding young data scientist could actually achieve higher than a 60% accuracy at the kaggle competition? Yes, they were young and naive enough to think just that. The researchers are embarrassed by how obsessed they became with this "challenge" and they are even more embarrassed by the amount of time they sunk into this endeavor. The husband of the researchers commented that this is not for naught -- this is all part of the greater "learning experience" and the things the researchers tried on this particular attempt will continue to fuel their future endeavors, if only as a guide for "what not to do." The researchers continue to feel like they might be heroines in their own adventures, just to one day be "discovered' as the unique, slightly obsessed, clearly oddly wired, quasi-intellectuals that will one day crack The Code, despite not even knowing that Code it is they are supposed to crack. Professor Gates, if you're reading this, please help me help myself -- what do you do when you go down such a deep rabbit hole that you've forgotten the taste of your own tongue?

Conclusion

There are still things that humans do better than machines. We know that horseradish doesn't belong in brownie recipes, we can tell if a tweet is sarcastic, we can identify if this is a photo of a chiuaua or a muffin. The reason machines can't do this yet isn't because they aren't "smart enough." Well, maybe it is, depending on your definition of intelligence. If intelligence is measured as the collection of everything we've ever learned, then yes, this is because the computer isn't "smart enough."

The computers simply haven't been given enough data to determine that those are blueberries and not chiuaua eyeballs. In the same way that a small child thinks every four-legged creature is "doggie" until s/he has been out in the world long enough to collect more data ("This four-legged creature is always bigger than a dog and it also makes a totally different noise!! I've noticed that the adults refer to this one as 'horse"' — "This four-legged creature isn't nearly as friendly but absolutely loves to snuggle, the adults refer to it as 'cat' so maybe it also isn't a "doggie!!") the computer is simply at a data-disadvantage.

The solution, just like with the small child, is expose the computer/child to more data. That is exactly what Amazon Mechanical Turk is doing. It is using "Artificial" Artificial Intelligence (humans) to feed the computers the data it will eventually need to be able to consistently say "chihuahua" vs "muffin."

[1] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115–124.

[2] Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng and Chris Potts. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).