How does the Qualitative Sentiment Influence the Quantitative Rating?

Group 10: Peter Ayral, Kaden Nguyen, Nathan Ng, Yanjie Qi

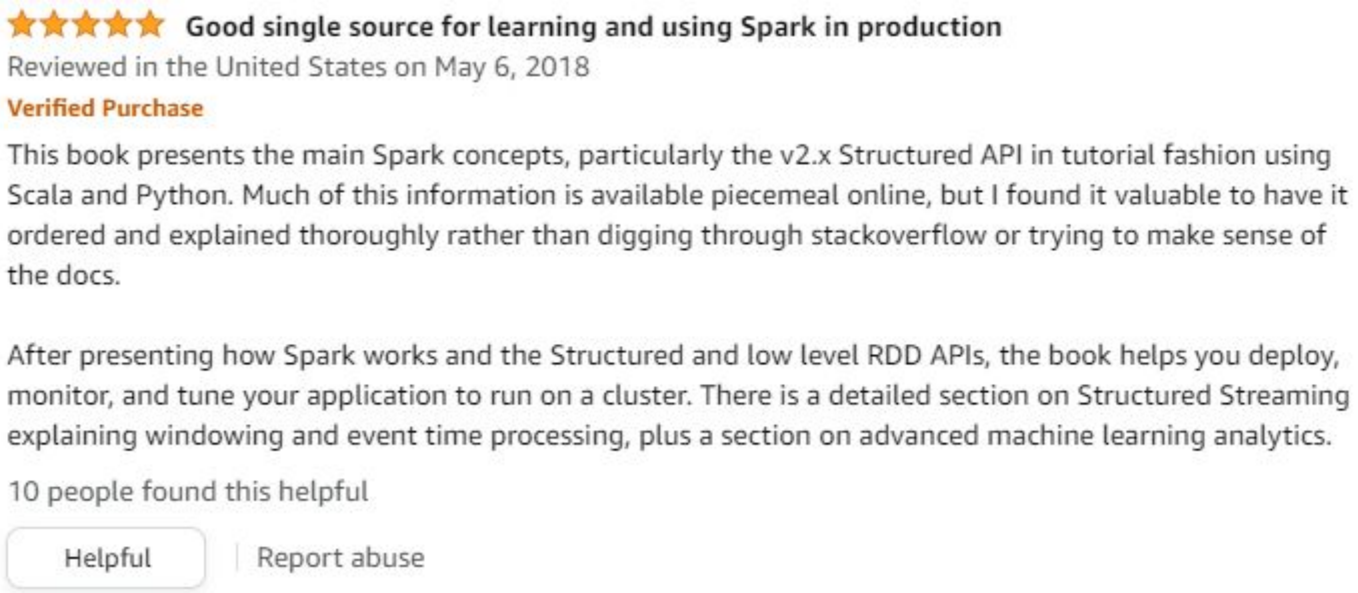
Executive Summary

- We wanted to practice sentiment analysis on big data. We decided to analyze reviews of books on kindle to predict if the sentiment was positive or negative.
- This is useful because it allows a publishing company to address negative feedback instantly.
- We categorized a positive sentiment as a rating of 3 or above, and a negative sentiment as a rating of below 3.
- We built pipelines and used feature extraction to preprocess the dataset.
- We used Linear and Logistic regression, Naive bayes, and a Random Forest model to predict this, and were happy with the results.

Data Summary

- Our data originates from Kaggle and more specifically, kindle’s marketplace. Each review has all the information from a basic review. It has over 200,000 rows and is over 600 MB. The most important columns were:
- **OverallRating**: The rating out of 5 that the reviewer left
- **Helpful**: How many people found the review helpful (and how many didn’t)
- **ReviewText**: The text review that the reviewer left
- **Summary**: The header for the review text

Example



★★★★★ **Good single source for learning and using Spark in production**

Reviewed in the United States on May 6, 2018

Verified Purchase

This book presents the main Spark concepts, particularly the v2.x Structured API in tutorial fashion using Scala and Python. Much of this information is available piecemeal online, but I found it valuable to have it ordered and explained thoroughly rather than digging through stackoverflow or trying to make sense of the docs.

After presenting how Spark works and the Structured and low level RDD APIs, the book helps you deploy, monitor, and tune your application to run on a cluster. There is a detailed section on Structured Streaming explaining windowing and event time processing, plus a section on advanced machine learning analytics.

10 people found this helpful

Preprocessing

- Dropped any null values
- Removed stop words from the text
- Removed symbols (&%”) from the text
- Show successful case1 where each asin is unique. This will allow them to be distinct between train and test
- The clean dataset had a total of 14880 rows used for training and test
- Pipeline1: Tokenizer, StringIndexer, OneHotEncoder, Vector Assembler, HashingTF

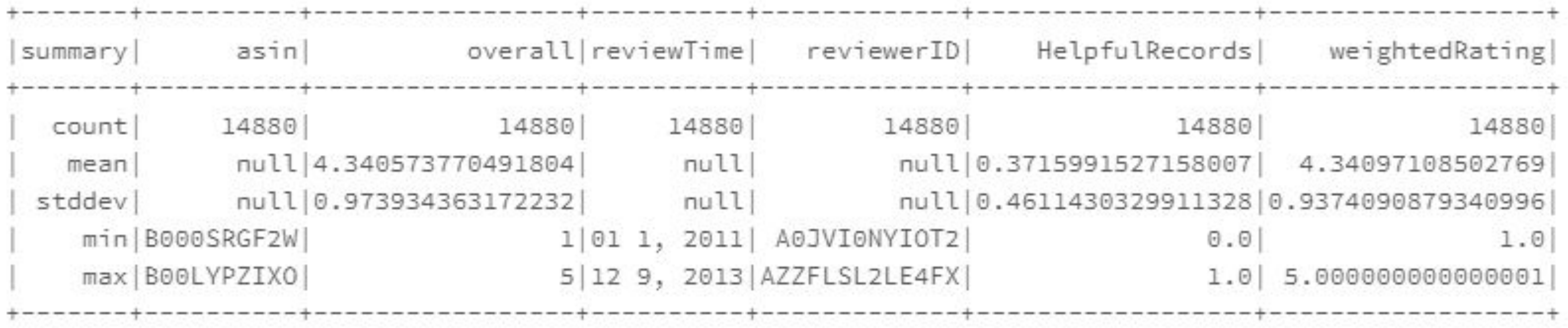
Feature Engineering: Weighted Rating

Two Situations:

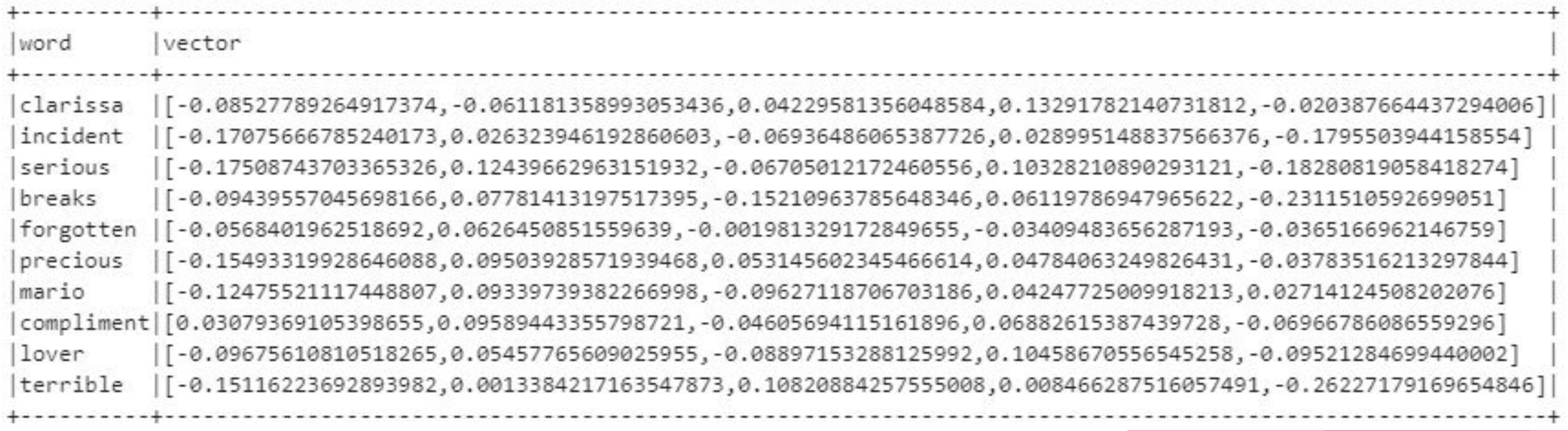
1. If there is no records of helpful, the rating will not be given the weight. (By Default, the weight is 1)
2. If there are records, then the weight will be depend on its proportion of records, specifically 1 + (records think the review is helpful) / (all records)

Groupby the asin (product) column, calculate the weight*rating and obtain the sum, and then divided by the sum of weights, now we have the new column WeightedRating as a group feature.

Exploratory Data Analysis

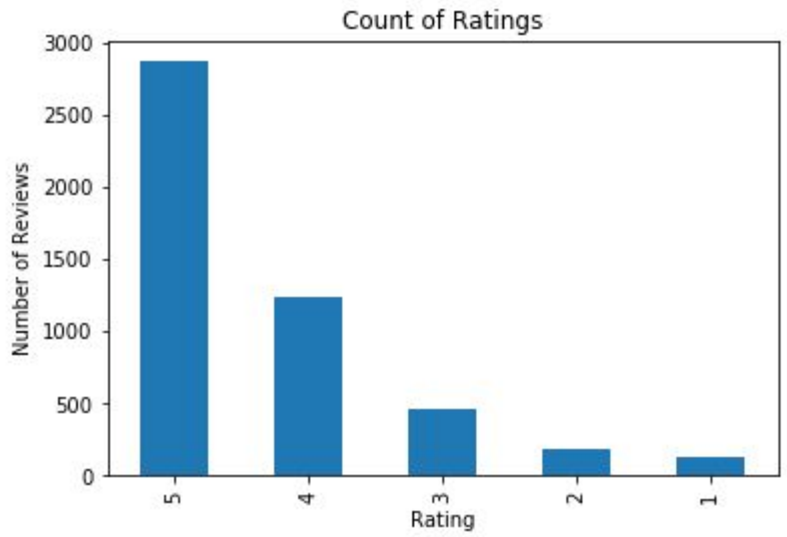


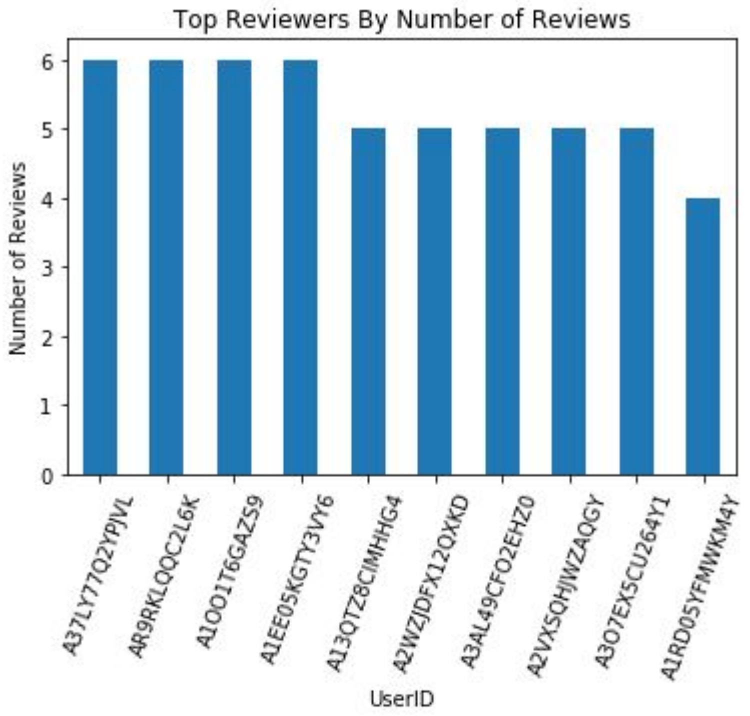
```
+-------+----------+-----------------+----------+------------+------------------+------------------+
|summary|      asin|          overall|reviewTime|  reviewerID|    HelpfulRecords|    weightedRating|
+-------+----------+-----------------+----------+------------+------------------+------------------+
|  count|     14880|            14880|     14880|       14880|             14880|             14880|
|   mean|      null|4.340573770491804|      null|        null|0.3715991527158007|  4.34097108502769|
| stddev|      null|0.973934363172232|      null|        null|0.4611430329911328|0.9374090879340996|
|    min|B000SRGF2W|                1|01 1, 2011| A0JVI0NYIOT2|               0.0|               1.0|
|    max|B00LYPZIXO|                5|12 9, 2013|AZZFLSL2LE4FX|               1.0| 5.000000000000001|
+-------+----------+-----------------+----------+------------+------------------+------------------+
```



```
+----------+----------------------------------------------------------------------------------------------------------+
|word      |vector                                                                                                    |
+----------+----------------------------------------------------------------------------------------------------------+
|clarissa  |[-0.08527789264917374,-0.061181358993053436,0.04229581356048584,0.13291782140731812,-0.020387664437294006]|
|incident  |[-0.1707566678524017,0.026323946192860603,-0.06936486065387726,0.028995148837566376,-0.1795503944158554]  |
|serious   |[-0.17508743703365326,0.12439662963151932,-0.06705012172460556,0.10328210890293121,-0.18280819058418274]  |
|breaks    |[-0.09439557045698166,0.07781413197517395,-0.15210963785648346,0.06119786947965622,-0.2311510592699051]   |
|forgotten |[-0.0568401962518692,0.0626450851559639,-0.001981329172849655,-0.03409483656287193,-0.0365166962146759]   |
|precious  |[-0.15493319928646088,0.09503928571939468,0.053145602345466614,0.04784063249826431,-0.03783516213297844]  |
|mario     |[-0.12475521117448807,0.09339739382266998,-0.09627118706703186,0.04247725009918213,0.02714124508202076]   |
|compliment|[0.03079369105398655,0.09589443355798721,-0.04605694115161896,0.06882615387439728,-0.06966786086559296]   |
|lover     |[-0.09675610810518265,0.05457765609025955,-0.08897153288125992,0.10458670556545258,-0.09521284699440002]  |
|terrible  |[-0.15116223692893982,0.0013384217163547873,0.10820884257555008,0.008466287516057491,-0.26227179169654846]|
+----------+----------------------------------------------------------------------------------------------------------+
```

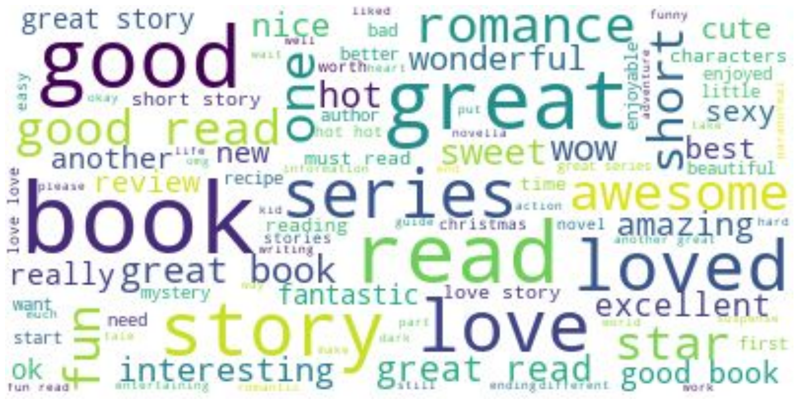
Bar Graphs


Count of Ratings
Number of Reviews
3000
2500
2000
1500
1000
500
0
5
4
3
2
1
Rating

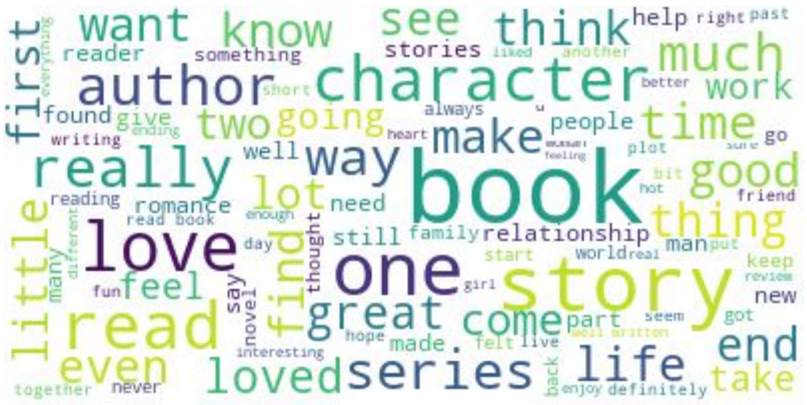

Top Reviewers By Number of Reviews
Number of Reviews
6
5
4
3
2
1
0
A37LY77Q2YPJVL
AR9RKLQOC2L6K
A1OO1T6GAZS9
A1EE05KGTY3VY6
A13QTZ8CIMHHG4
A2WZJDFX12QXKD
A3AL49CFO2EHZ0
A2VXSQHJWZAQGY
A3O7EX5CU264Y1
A1RD05YFMWKM4Y
UserID

Word Clouds

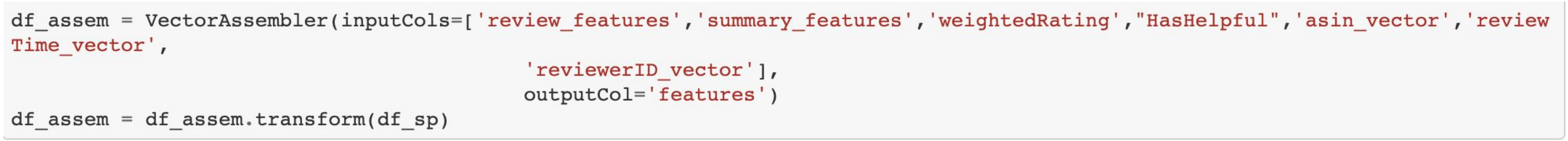
Summary



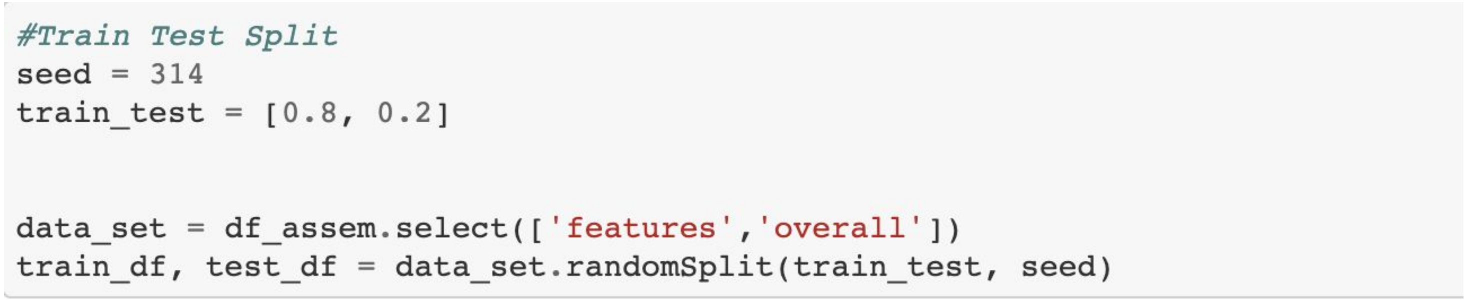
Reviews



Sampling



```
df_assem = VectorAssembler(inputCols=['review_features','summary_features','weightedRating',"HasHelpful",'asin_vector','review
Time_vector',
                                          'reviewerID_vector'],
                                          outputCol='features')
df_assem = df_assem.transform(df_sp)
```



```
#Train Test Split
seed = 314
train_test = [0.8, 0.2]


data_set = df_assem.select(['features','overall'])
train_df, test_df = data_set.randomSplit(train_test, seed)
```

Variables

Key Predictors:

- reviewTime, reviewText, helpful

Target Response:

- Overall: the rating given by Amazon customers. Possibilities include 0-5

Models Used

- Linear Regression (Benchmark Model)
- Logistic Regression via Pipeline1
- Naive Bayes via Pipeline1
- Random Forest

Linear Regression (Benchmark Model)

- Models the relationship between two variables by fitting a linear equation to observed data
- Advantages: simple to implement, easily interpretable, great when you know relationship between independent and dependent variable is linear
- Disadvantages: outliers greatly affect the model, assumes linear relationship between dependent and independent variables, therefore can only represent linear relationships
- RMSE of Training Data = 0.245
- RMSE of Test Data = 0.255

Logistic Regression

- Used for binary classification
- Advantages: efficient, highly interpretable, doesn’t require any tuning, outputs well-calibrated predicted probabilities
- Disadvantages: can’t solve nonlinear problems, only important relevant features should be used

Test Accuracy	Test Recall	Test Precision	F1 Score
0.927	1.0	0.933	0.966

Naive Bayes

- Classification technique based on Bayes' Theorem with an assumption of independence between predictors
- Advantages: handles both continuous and discrete data, not sensitive to irrelevant features, doesn't require that much training data
- Disadvantages: independence between all predictors rarely happens in real life, zero frequency

Test Accuracy	Test Recall	Test Precision	F1 Score
0.930	1.0	0.930	0.964

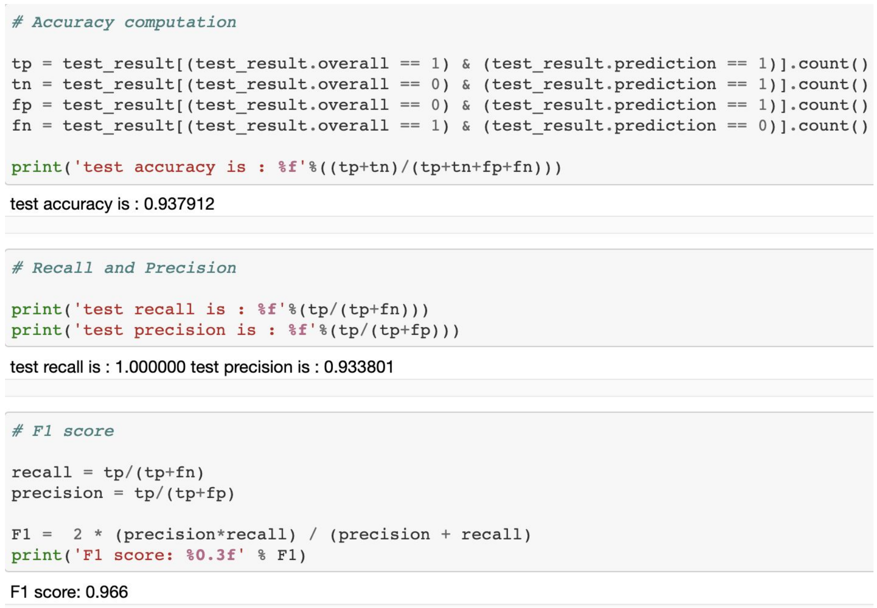
Random Forest

- Builds multiple decision trees and merges them together to create a more accurate and stable prediction
- Advantages: used for regression and classification tasks, easy to view relative importance assigned to the features
- Disadvantages: large number of trees can make the algorithm slow and ineffective for real-time predictions

Test Accuracy	Test Recall	Test Precision	F1 Score
0.930	1.0	0.930	0.964

Evaluation

- Test Accuracy, Test Recall, Test Precision, and F1 Score



```
# Accuracy computation

tp = test_result[(test_result.overall == 1) & (test_result.prediction == 1)].count()
tn = test_result[(test_result.overall == 0) & (test_result.prediction == 1)].count()
fp = test_result[(test_result.overall == 0) & (test_result.prediction == 1)].count()
fn = test_result[(test_result.overall == 1) & (test_result.prediction == 0)].count()

print('test accuracy is : %f'%((tp+tn)/(tp+tn+fp+fn)))
```

```
test accuracy is : 0.937912
```

```
# Recall and Precision

print('test recall is : %f'%(tp/(tp+fn)))
print('test precision is : %f'%(tp/(tp+fp)))
```

```
test recall is : 1.000000 test precision is : 0.933801
```

```
# F1 score

recall = tp/(tp+fn)
precision = tp/(tp+fp)

F1 =  2 * (precision*recall) / (precision + recall)
print('F1 score: %0.3f' % F1)
```

```
F1 score: 0.966
```

Model Comparison

	Test Accuracy	Test Recall	Test Precision	F1 Score
Logistic Regression	0.927	1.0	0.933	0.966
Naive Bayes	0.930	1.0	0.930	0.964
Random Forest	0.930	1.0	0.930	0.964

Conclusion

Naive Bayes is the best model. It has the lowest test error

The test error difference between the baseline model and the Naive Bayes is .46

Future Work and Problems Encountered

- Future Work
  - “Case 2”: Use all the instances of ASIN, instead of just unique ASIN’s
  - Use unsupervised learning
  - Can further examine for potential interactions in the Linear Regression Context

- Problems Encountered
  - Using Vector Assembler vs Labeled Points
  - Issues with connecting to the server

Thank you!