





Individual



Predictive Model

CDs and Vinyls
Siby Suriyan



Goal



The goal of this submission is to try to improve the predictability of a model for the awesomeness/not-awesomeness of a product. Researching into the best methods to do that, I determined that one of the most important features is sentiment analysis on the summary and reviewText of each review. Manual review of summary and reviewText and their associated sentiment scores, I discovered that some of the reviews had inaccurate sentiment scores in relation to the summary and reviewText . So, I researched into better sentiment analysis tools.



Another strategy for improving predictability is to try new binary classification models that we went over in class. Since the last submission, we have learned about neural networks and KNN. So I experimented with those. I also experimented with decision trees more. My group did not spend too much time with decision trees in previous submissions, instead opting to work with random forests.

Goal



My last goal of the project is to create, test, and predict on a model that is within my computing resources. In previous submissions, my group used my partner's powerful desktop to fit and test models. I have much less computing power. My model needs to run efficiently on my computer while still being accurate. Often, the more complex, powerful, and accurate a model is, the more computing resources it requires to run. This is my primary reason to experiment more with decision trees; they are less resource heavy than random forests, which may not run on my computer.



New Approaches



1. **Flair Sentiment Analysis**- Upon further research into sentiment analysis tools, I discovered that Flair performed better than Kaludi/Hugging Face and Vader. We still experienced the same issues with Kaludi that were present in Vader. Kaludi would incorrectly score positive text and summary reviews to be negative and vice versa.
2. **Neural Networks**- A system of algorithms and components that understands patterns in data through multiple layers of neuron. After all the data has passed through all the layers, and a prediction has been made, back propagation and gradient descent takes place, to minimize the error between the predictions and ground truths.
3. **K Nearest Neighbors**- A clustering algorithm, where training data is clustered together based on Euclidean distance (most often) and labelled based on groups of clusters. Then, sample data is labelled based on which cluster they belong to, which is also found with Euclidean distance (most often).
4. **Decision Tree**- While Decision Trees are not necessarily new, my group has not invested too much time experimenting with them. We have not done Hyperparameter grid search or bagging with it. Decision Trees work by splitting the features in the dataset along conditions of the values in a feature.



Flair vs. Kaludi



I decided to research into more sentiment analysis tools after manual review of the sentiment scores with the reviews. Upon inspection, I saw that Kaludi would give positive sentiment scores to reviews that had a 0 for awesomeness and vice versa. I believe this is because Kaludi has a difficult time in determining sentiment for longer reviews that are not directly related to the product. For example, from my external research, I found that Kaludi scores this review negatively: *"I was so glad Amazon carried these batteries. I have a hard time finding them elsewhere because they are such a unique size. I need them for my garage door opener. Great deal for the price."* Meanwhile, Flair scores it positively. In more nuanced reviews, Flair outperforms Kaludi. Flair was developed with a neural network and was trained on IMDb movie reviews.

Flair & Kaludi's F1 scores with same features and model with 10 cross fold val:



Flair- 0.7480

Kaludi- 0.6693

New Models + Hyperparameter GS



KNN

Optimal Parameters:

Neighbors = 200
Weights = distance

F1 Score: 0.653

These F1 scores were found with 3 cross-fold validation for quickly finding best parameters. F1s with 10 cross-fold validation are on next slide.

Neural Nets

Optimal Parameters:

Activation: tanh
Alpha: 0.01
Hidden Layer Sizes: 128,
128, 128
Learning Rate: adaptive
solver: adam

F1 = 0.875

Decision Tree

Optimal Parameters:

Max Depth: 10
Min Samples Split: 10
Min Samples Leaf: 5
Max Features: 0.7

F1 = 0.7225

Model's Scores w/ Flair

Our the final team submission, the best performing models were Logistic Regression, SVM, and Random Forest. We tested these models (plus the new models) with 10 cross-fold validation, Flair sentiment, and the best parameters from previous submission's best parameters and the previous slide's best parameters.

01 **Logistic Regression**
F1 = 0.7399

02 **Random Forest**
F1 = 0.7471

03 **KNN**
F1 = 0.655

04 **SVM**
*compute time took too long

05 **Neural Nets**
F1 = 0.7038

06 **Decision Tree**
F1 = 0.7217



Model's Scores w/ Flair + Bagging

01 Logistic Regression
F1 = 0.7405

02 Random Forest*
F1 = 0.7138

03 KNN
F1 = 0.6663

04 SVM
*compute time took too long

05 Neural Nets
*compute time took too long

06 Decision Tree
F1 = 0.7529

Random Forests inherently use bagging, so applying extra bagging to RF would not help

Model Scores with Flair + AdaBoost

01 Logistic Regression
F1 = 0.7331

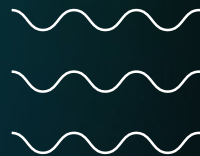
02 Random Forest
*compute time took too long

03 KNN
*not applicable

04 SVM
*not applicable

05 Neural Nets
*compute time took too long

06 Decision Tree
F1 = 0.6849



Winner: Decision Tree + Flair + Bagging



After extensive testing within the confines of my computing resources, I found Decision Trees with Bagging performed the best with Flair sentiment analysis.

Decision Trees are a machine learning algorithm that trains on data by splitting features in the dataset along branches to create the most homogenous leaf nodes possible with respect to the target-value feature. Each node contains a specific feature from which the data will be split, so branches from said node will be conditions for splitting data. The Bagging additive improves the model's accuracy by splitting the data into many subsets and training a decision tree on each subset, creating a final model with multiple trees. For predictions, each tree votes on the classification of a testing sample, and majority vote wins.



During testing the Decision Tree in our previous submission, we performed hyper-parameter grid search to find the best parameters to train our data. Those parameters were also used in this model. Ultimately, I was unable to test Random Forest + Flair + AdaBoost because a lack of computing resources. AdaBoost is compute heavy.

Hyperparameter GS (previous submission)



Logistic Regression

F1 Score= 0.691

Optimal Parameters:
C = 4.28

Random Forest

F1 Score= 0.698

Optimal Parameters:
Max Depth = 9
Max Features = 0.7
N Estimators = 185
Max Samples = 0.8

SVM

F1 Score: 0.697

Optimal Parameters:
Kernel = rbf
C = 0.1
Gamma = 1



Improvements

Based on the final team submission, we can see major improvements from using the Flair sentiment analysis compared to the Kaludi/Hugging Face sentiment analysis. I have summarized the improvements below. I have included the highest f1 score of each model with the various additive associated with their best f1 score with Kaludi/Hugging Face and Flair. I saw a 0.499 increase between the best models across submissions.

01

Logistic Regression

F1 = 0.684 -> 0.7405

02

Random Forest

F1 = 0.703 -> 0.7471

03

KNN

***not tried with Kaludi/Hugging Face**

04

SVM

***compute time took too long (with flair)**

05

Neural Nets

***not tried with Kaludi/Hugging Face**

06

Decision Tree

F1 = 0.672 -> 0.7529

Model Scores (previous submission)

AdaBoost

Bagging

We took the best parameters found for each classifier and applied them to bagging and adaboost. The results those two optimizations follow with 10 cross fold validation.

Logistic Regression

F1 Score= 0.684

Logistic Regression

F1 Score= 0.621

Random Forest

F1 Score= 0.703

Random Forest

F1 Score = 0.688

SVM

*not applicable

SVM

F1 Score = 0.6768



Score of the decision tree with Kaludi/Hugging Face

```
[ ] # Decision Tree with Kaludi/Hugging Face
clf = BaggingClassifier(DecisionTreeClassifier(max_depth = 10, max_features = .7, min_samples_leaf = 5, min_samples_split = 10), max_samples = 0.25, max_features = 1.0, n_estimators =
f1_scores = cross_val_score(clf, old_X, old_y, cv=10, scoring="f1", n_jobs=-1, verbose = 1)

print(f1_scores)
print(np.mean(f1_scores))

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done 6 out of 10 | elapsed: 2.6min remaining: 1.8min
[0.66100403 0.66048912 0.66172428 0.68893898 0.71780632 0.67954502
 0.65235851 0.67557339 0.66303344 0.65438897]
0.6714862062469984
[Parallel(n_jobs=-1)]: Done 10 out of 10 | elapsed: 4.2min finished
```

What I Learned



I learned that it is important to not dismiss certain classification models because they are very simple. This was the original reason why my group dismissed decision trees; they were a simple model. But, in the end, decision trees ended up performing the best for predicting the awesomeness/not-awesomeness of a product.

I also learned that the feature vector in a training set is more important to accurately train a model than using a complex model. I was able to see better improvements in F1 score by creating a more detailed feature vector than by using different models. In terms of a good feature vector, in addition to including the mean sentiment of text and summary of each product, I should also include the standard deviation sentiment of text and summary. And I should give greater weight to reviews with lots of votes and verifications.

Also, I learned that Bagging is more effective and quicker than Boosting. In the previous submission, my team focused heavily on boosting; however, in my findings, Bagging helped f1 scores more.



Last, I learned the dangers of overfitting. When I was originally training and testing my models, I was not performing any cross validation. As a result, my neural net model had a f1 of 0.875, the highest of any model. However, after doing 10 cross-fold validation, my decision tree + bagging model had the highest score.