# K-Neighbors Classifier of Reviews

Miner user: lalvarez

Rank: 54

Accuracy: 0.74

## My approach

### Pre-processing of the data

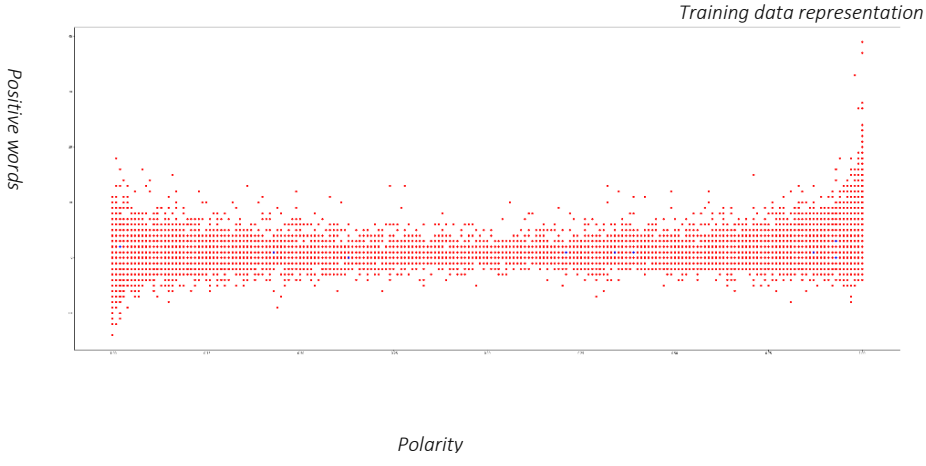
This program analyzes every review so we can get as much information as possible. Reviews are cleaned by eliminating all the punctuation marks.
After that, we get the useful words from every review by ignoring the padding words and the special characters.
Now that the document is clean, we can start analyzing the sentiments of the reviews, for that task I have combined two different methods:

[('good', 11682), ('great', 7692), ('bad', 6805), ('much', 6504), ('many', 6363), ('little', 5870), ('best', 4480), ('real', 4024), ('new', 3984), ('first', 3858), ('old', 3585), ('young', 3288), ('big', 3078), ('whole', 2805), ('last', 2690), ('original', 2646), ('least', 2464), ('it.', 2418), ('/>the', 2243), ('main', 2223), ('funny', 2200), ('different', 2063), ('worst', 1980), ('sure', 1902), ('hard', 1879), ('American', 1828), ('special', 1805), ('true', 1787), ('high', 1690), ('black', 1641), ('nice', 1597), ('better', 1591), ('second', 1576), ('full', 1550), ('beautiful', 1544), ('poor', 1531), ('small', 1467), ('entire', 1456), ('excellent', 1416), ('right', 1411), ('several', 1387), ('worth', 1358), ('film.', 1350), ('short', 1322), ('human', 1318), ('wonderful', 1293), ('classic', 1286), ('it,', 1250), ('enough', 1250), ("i'm", 1237), ('able', 1230), ('final', 1228), ('next', 1210), ('give', 1189), ('early', 1173), ('interesting', 1154), ('long', 1142), ('dead', 1139), ('film,', 1133), ('low', 1128), ('stupid', 1104), ('white', 1098), ('top', 1077), ('terrible', 1032), ('movie.', 981), ('fine', 949), ('wrong', 940), ('perfect', 936), ('complete', 927), ('strong', 917), ('favorite', 911), ('awful', 907), ('obvious', 903), ('huge', 890), ('live', 885), ('major', 884), ('local', 863), ('single', 846), ('due', 838), ('British', 831), ('musical', 815), ('decent', 809), ('serious', 799), ('#eofthis', 793), ('worse', 785), ('actual', 783), ('important', 779), ('horrible', 778), ('similar', 777), ('modern', 777), ('comic', 756), ('usual', 752), ('romantic', 736), ('certain', 735), ('typical', 733), ('English', 730), ('evil', 716), ('less', 707), ('overall', 703), ('happy', 693), ("can't", 692), ('greatest', 689), ('French', 685), ('famous', 681), ('strange', 680), ('movie,', 671), ('late', 662), ('wish', 660), ('simple', 658), ('sexual', 655), ('screen', 651), ('sad', 650), ('slow', 649), ('difficult', 645), ('brilliant', 640), ('clear', 633), ('Japanese', 633), ('easy', 628), ('possible', 623), ('cheap', 614), ('previous', 614), ('emotional', 614), ('third', 604), ('"the', 601), ('various', 597), ('scary', 595), ('ridiculous', 594), ('female', 593), ('personal', 592), ("that's", 588), ('enjoyable', 585), ('dark', 576), ('hilarious', 566), ('shot', 564), ('particular', 561), ('weak', 559), ('fantastic', 559), ("there's", 558), ('dramatic', 556), ('political', 555), ('interested', 552), ('know', 544), ('red', 543), ('general', 540), ('close', 538), ('total', 537), ('middle', 529), ('social', 525), ('up.', 522), ('see', 520)]

*The first coordinate* will represent the review's polarity. I have used a method from the nltk library (SentimentIntensityAnalyzer). This will return a value that represent the positive, negative or neutral feeling of the review.

*The second coordinate* states the overall of positive and negative words in a review. I tried to make this as precise as possible so I created an auxiliary program to find the most common adjectives in the 25,000 reviews. The picture above shows the result of my research, with the most common adjectives and the number of times they appear in the reviews.
For each positive word the review gets one positive point, whereas each negative word gives one negative point to the review.

*Training data representation*



## Predicting test label

Once the model is created, we proceed to analyze the test data obtaining these characteristics from the test data, finding the distance between the test point and all of the other points of my training data. The program sorts the distances and chooses the K smallest values.
Finally, we choose the label of the KNN's mode and label this test point with that mode value.

After trying different techniques and approaches, I found that I get better results when using a higher number of neighbors (bigger than or equal 50) whereas if I increase that number to 100 the accuracy stayed the same.

## Accuracy

The accuracy gives us an idea of how good is our classifier. For this project, the accuracy is computed using the next formula:

$$Accuracy = \frac{Correct\ classifications}{Total\ number\ of\ reviews}$$

Imagine an example where we have an imbalanced dataset, many objects of one class 99% and 1% of another. If we always predict the same class, we would be able to get a 99% accuracy. Therefore, the classifier will be good for this dataset, but useless if we use another dataset with more spread values or with a larger percent of values labeled with the second class. In this case, the accuracy value can be misleading. It is really high, making us think that we made a good model, which will be able to classify all my data correctly anytime, when this model is not good indeed. We can use another measures that would be more accurate, for example, F1 score and Brier score.

## Efficiency

To improve efficiency we can use different tools for example PCA that would reduce the number of dimensions since my approach already has 2 dimensions I decide to not use this technique and instead of that to improve the efficiency of my program I decided to apply another design decisions:

- After analyzing my data and running my program several times I realized that the third component I used to have (length) was not representative so I reduce the dimensionality of my data from 3 dimensions to 2 dimensions keeping just the most representative parameters (polarity and average words).

- Another decision of design I took was to change my first approach using the cosine similarity to find the k nearest neighbors. I decided to use the Euclidean distance due to the results I obtained were better and the computing time were reduced considerably.

- The value k has been chosen carefully. I computed my program with different values for this variable and, I find out that there is a threshold, the value 25. The accuracy level increases if we use a higher number for k, until we reach this value. After that, the accuracy maintains a constant value no matter how much we increase the number of neighbors.
  In this picture, you can see the function of the accuracy and the k value.