AN EXPLORATION OF LINEAR ALGEBRA FOR MACHINE LEARNING

NICOLAS JORQUERA

OVERVIEW - DATA

book_id	review_id	rating	review_text	date_added	date_updated	read_at	started_at	n_votes	n_comments
18471619	66b2ba840f9bd36d6d27f46136fe4772	3	Sherlock Holmes and the Vampires of London \n ...	Thu Dec 05 10:44:25 -0800 2013	Thu Dec 05 10:45:15 -0800 2013	Tue Nov 05 00:00:00 -0800 2013		0	0
6315584	72f1229aba5a88f9e72f0dcdc007dd22	4	I've never really liked Spider-Man. I am, howe...	Wed Aug 10 06:06:48 -0700 2016	Fri Aug 12 08:49:54 -0700 2016	Fri Aug 12 08:49:54 -0700 2016	Wed Aug 10 00:00:00 -0700 2016	0	0

Dataset 1 - Reviews of Comic Books by Individuals

	title	book_id	text_reviews_count	average_rating	ratings_count
28	A Game of Thrones: The Graphic Novel, Vol. 4	22889758	71	4.09	637
43	Gambit, Volume 2: Tombstone Blues	17277799	34	3.75	370
46	Ultimate Comics: Spider-Man, by Brian Michael ...	17277795	71	4.10	651
68	Orphan Black #1	25051311	50	3.42	303
74	Space Battle Lunchtime Volume 2: A Recipe for ...	33590271	26	3.99	155

Dataset 2 – Information on Books from Dataset 1

DIMENSIONALITY REDUCTION TECHNIQUES

Throughout this semester we were introduced to a variety of Dimensionality Reduction techniques. In this section the benefits of 2 techniques are shown by utilizing the datasets mentioned

1. **Singular Value Decomposition** will be used to create a Collaborative Filtering Recommendation System. This becomes useful when there is a lot of sparce data; and is a more robust matrix factorization technique. We will also explore how to utilize SVD from the sci-kit learn library.
2. **Principal Component Analysis** will be used to reduce the data into 2 components and try to find any patterns that emerge from the 2-dimensional representation of the data.

DATA PROCESSING

Setting restrictions to number of reviews left -

```
# Trying to restrict to users with more than ___ reviews. - In this case 10
user_count = ratings[['user_id','book_id']].groupby('user_id').count()
user_count = user_count[user_count['book_id'] >= 10]
```

```
# Reducing the overall ratings matrix to include books/users with more than 10 reviews.
ratings = ratings[ratings["user_id"].isin(user_count.index) & ratings["book_id"].isin(book_count.index)]
```

Creating a Sparce Matrix -

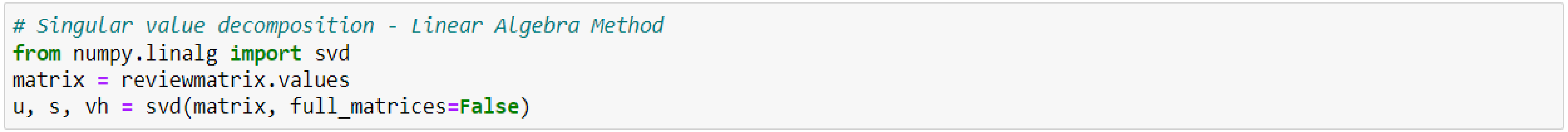
```
reviewmatrix = ratings.pivot(index="user_id", columns="book_id", values="rating").fillna(0)
```

```
                         book_id  866  868  869  870  871  873  2879  2880  2881  2882  ...  35541317  35657815  35669801  35703417  35708105  35833506
                         user_id
0008931c0cde961e9c802c5a58196d23  0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0   0.0  ...       0.0       0.0       0.0       0.0       0.0       0.0
000efb30c5236d7437c3cdf4bf3e4dc7  0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0   0.0  ...       0.0       0.0       0.0       0.0       0.0       0.0
```

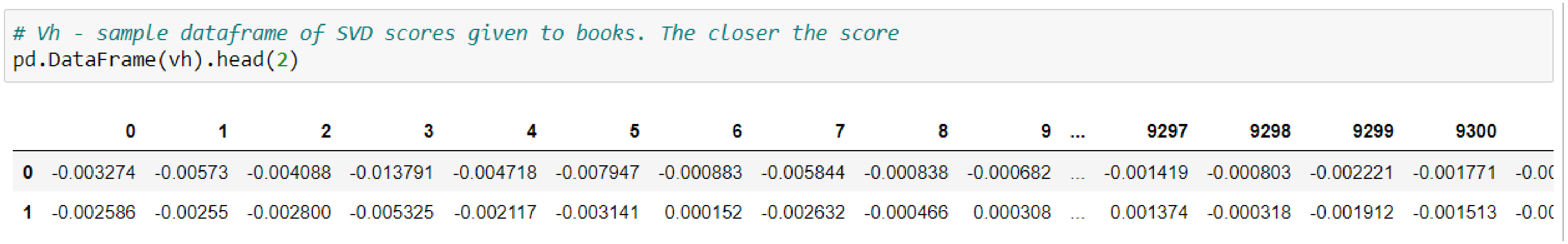
SINGULAR VALUE DECOMPOSITION – UTILIZING COSINE SIMILARITY

SVD is a type of matrix factorization that decomposes a matrix A into the components

- In the case of our matrix, the vh values are the right singular vectors of A that pertain the books, while the u values pertain to the people reviewing the books



```python
# Singular value decomposition - Linear Algebra Method
from numpy.linalg import svd
matrix = reviewmatrix.values
u, s, vh = svd(matrix, full_matrices=False)
```



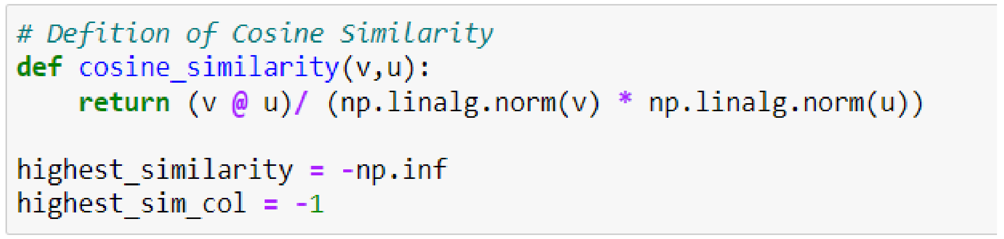
```python
# Vh - sample dataframe of SVD scores given to books. The closer the score
pd.DataFrame(vh).head(2)
```

	0	1	2	3	4	5	6	7	8	9	...	9297	9298	9299	9300	
0	-0.003274	-0.00573	-0.004088	-0.013791	-0.004718	-0.007947	-0.000883	-0.005844	-0.000838	-0.000682	...	-0.001419	-0.000803	-0.002221	-0.001771	-0.0
1	-0.002586	-0.00255	-0.002800	-0.005325	-0.002117	-0.003141	0.000152	-0.002632	-0.000466	0.000308	...	0.001374	-0.000318	-0.001912	-0.001513	-0.0

COSINE SIMILARITY

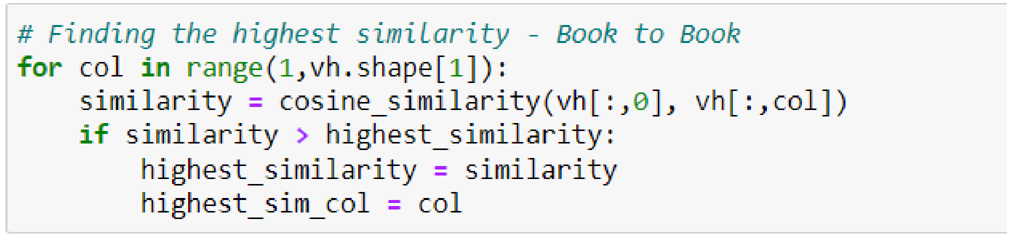
To determine which books are closest to one another we need to Cosine Similarity, which is a metric that measures how similar two vectors are in a multi-dimensional space. In this case, how close books are to one another.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

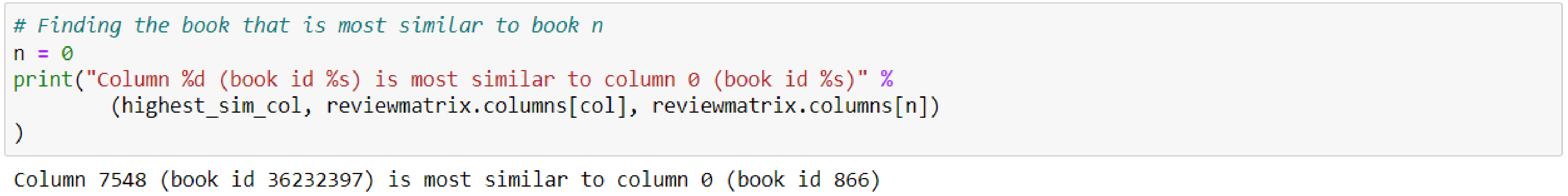


```python
# Defition of Cosine Similarity
def cosine_similarity(v,u):
    return (v @ u)/ (np.linalg.norm(v) * np.linalg.norm(u))

highest_similarity = -np.inf
highest_sim_col = -1
```



```python
# Finding the highest similarity - Book to Book
for col in range(1,vh.shape[1]):
    similarity = cosine_similarity(vh[:,0], vh[:,col])
    if similarity > highest_similarity:
        highest_similarity = similarity
        highest_sim_col = col
```

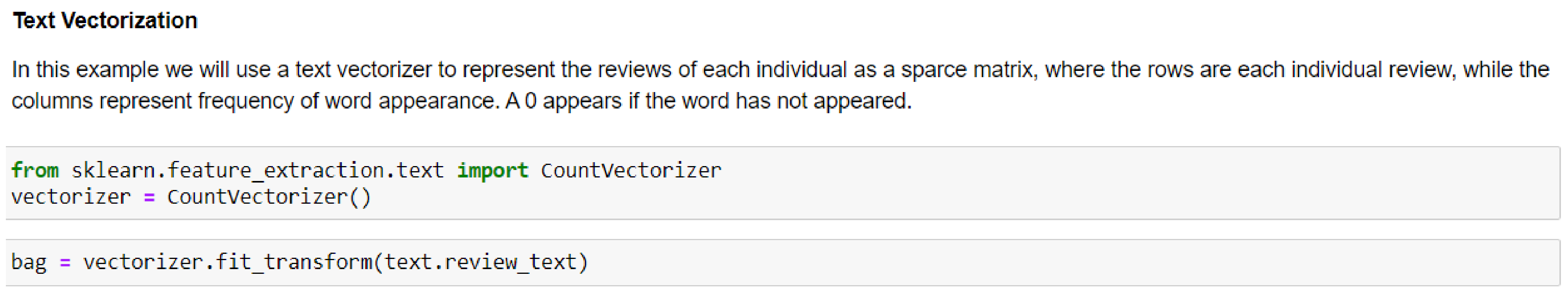


```python
# Finding the book that is most similar to book n
n = 0
print("Column %d (book id %s) is most similar to column 0 (book id %s)" %
        (highest_sim_col, reviewmatrix.columns[col], reviewmatrix.columns[n])
)
```

```
Column 7548 (book id 36232397) is most similar to column 0 (book id 866)
```

SINGULAR VALUE DECOMPOSITION – SCI-KIT LEARN

Here we utilize sci-kit learn, a powerful Machine Learning algorithm various tools we have been learning throughout the semester. However, this time we will implement SVD differently.



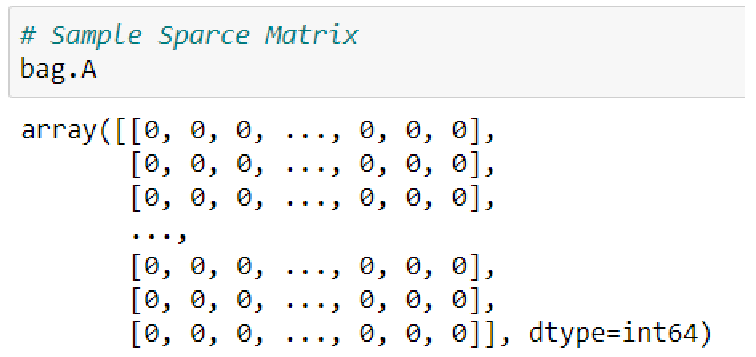
Text Vectorization

In this example we will use a text vectorizer to represent the reviews of each individual as a sparce matrix, where the rows are each individual review, while the columns represent frequency of word appearance. A 0 appears if the word has not appeared.

```python
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
```

```python
bag = vectorizer.fit_transform(text.review_text)
```

Here we can utilize another purpose of SVD, specifically to reduce dimensionality when working with ML problems. Fortunately, in this case the data is not significant, but if I were to utilize the full dataset of text to vectorized words the matrix can get very sparce!

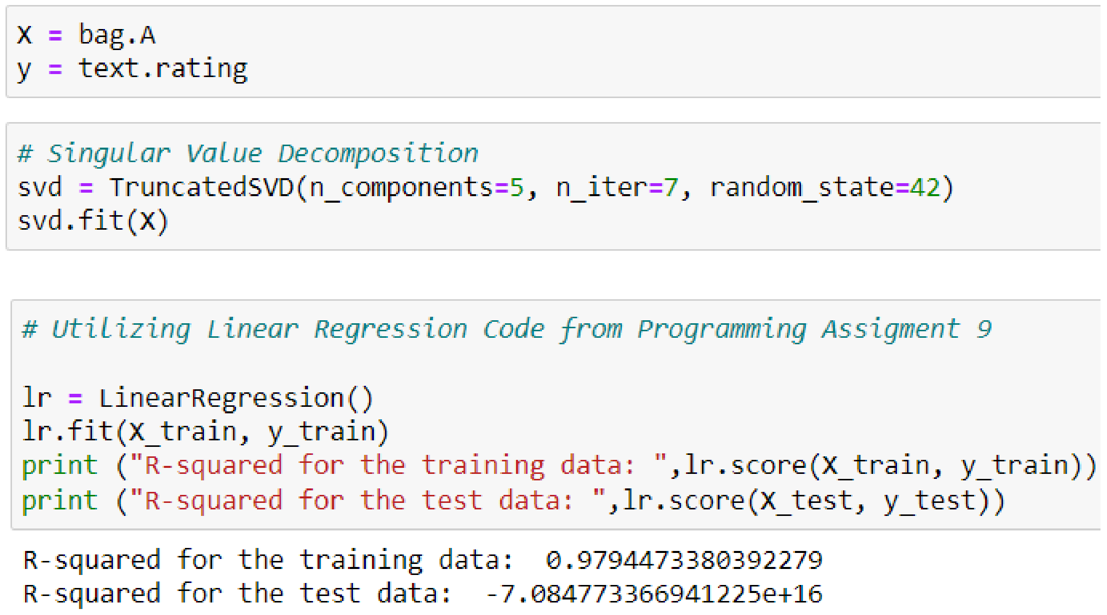


```python
# Sample Sparce Matrix
bag.A
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

LINEAR REGRESSION MODEL

Utilizing this we can create a machine learning model that can predict the score based on the review of the user. The independent values are the Truncated SVD text reviews while the dependent value is the rating.



```
X = bag.A
y = text.rating
```

```
# Singular Value Decomposition
svd = TruncatedSVD(n_components=5, n_iter=7, random_state=42)
svd.fit(X)
```

```
# Utilizing Linear Regression Code from Programming Assigment 9

lr = LinearRegression()
lr.fit(X_train, y_train)
print ("R-squared for the training data: ",lr.score(X_train, y_train))
print ("R-squared for the test data: ",lr.score(X_test, y_test))
```

```
R-squared for the training data:  0.9794473380392279
R-squared for the test data:  -7.084773366941225e+16
```

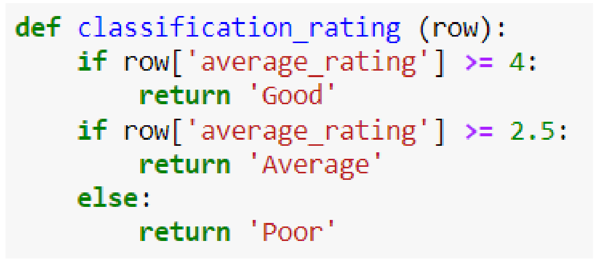
Observation:

Future improvements could be made to the model to make it more accurate, however I chose to focus on the Linear Algebra applications. Normally, Neural Networks are used for NLP problems. Furthermore, k-fold cross validation and other feature engineering techniques can also be utilized to optimize the cost function.

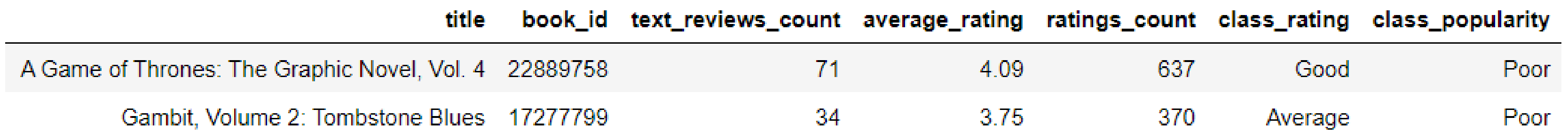
PRINCIPAL COMPONENT ANALYSIS – DATA ANALYSIS

- At this point we have reduced the dimensionality of the dataset into 2 components, PCA 1 and PCA 2. However, to have a good graphical sense if there is any correlations here, we want to have three different classifications for this data: good, average and poor.
- Because not enough information was provided; we will separate these books into different 'grades'. In this case these grades will be added as new columns to compare to PCA clustering. The cutoff point for the classification was determined using the statistical information of each column. (Comparison of the mean and standard deviation)

Implementing this:



```
def classification_rating (row):
    if row['average_rating'] >= 4:
        return 'Good'
    if row['average_rating'] >= 2.5:
        return 'Average'
    else:
        return 'Poor'
```



title	book_id	text_reviews_count	average_rating	ratings_count	class_rating	class_popularity
A Game of Thrones: The Graphic Novel, Vol. 4	22889758	71	4.09	637	Good	Poor
Gambit, Volume 2: Tombstone Blues	17277799	34	3.75	370	Average	Poor

IMPLEMENTING PCA

Principal Component Analysis Algorithm

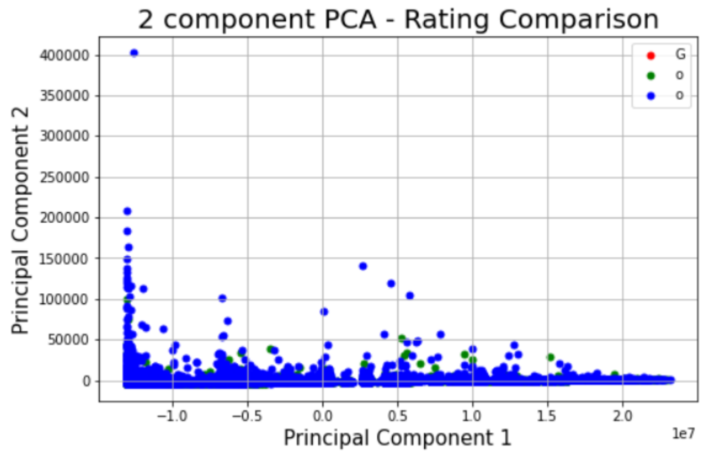
```
# Principal Component Analysis - 2 components were chosen; to depict them graphically.
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(books2)
```

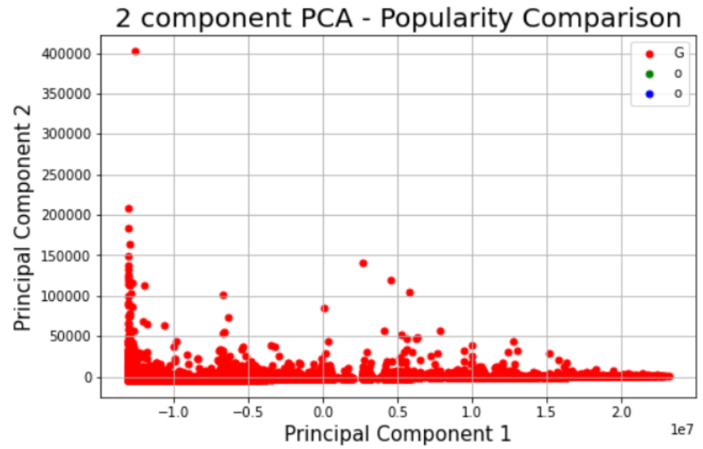
```
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

```
finalDf = pd.concat([principalDf, books.class_rating, books.class_popularity], axis = 1)
finalDf.head(2)
```

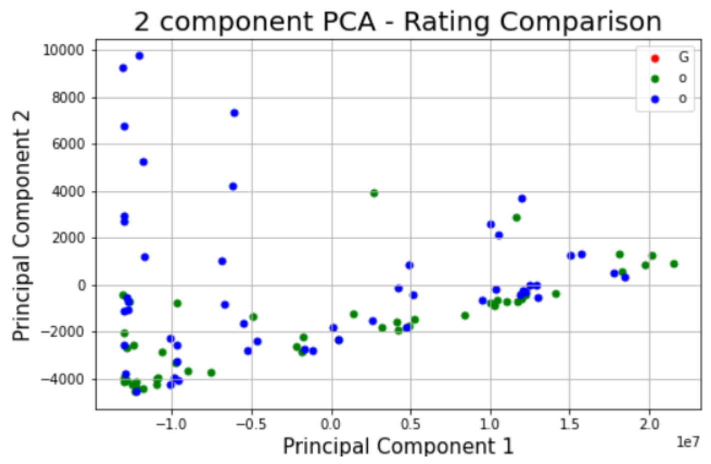
•Unfortunately, the plots below were not very conclusive. Although the first principal component accounts for almost 99% variance, there was not a clear distinction between data points above. Normally, PCA is a great way to visualize different classifications, however in this case this classification was self imposed by the user (me) and resulted in unsubstantial results.

Graphical Representation:





Reduced Sample Size of 100



CONCLUSION

In each of the situations I pointed out the limitations of the algorithm and what improvements could have been made to improve their performance. I hope to have shown how linear algebra is instrumental for the purposes of Machine Learning. However, each algorithm has its limitations, whether they be mathematical limitations or limitations imposed by the user due to the quality of the data. It is the responsibility of the data scientist to decide when to use which algorithm.