COGS 118B FINAL PROJECT: CUSTOMER SEGMENTATION

ALANNA MERLAN, YEWON HONG, ALEX BUMBALOV, DAVID THAI , MAXIM KONDRASHUK, NIL BESERLER

INTRODUCTION AND MOTIVATION

Today competition between businesses is higher than ever and data is a valuable resource for decision making.

Customer segmentation, allows businesses to determine strategies, and better target their customers. For example grouping their customers by their locations can allow a business to allocate more resources in a more dense location and focus on marketing for other locations. This was a topic that interested all of us since it's commonly used and very informative.

In this projects we use common characteristics to group the customers and retract valuable information for businesses. We accomplish this with K-means and PCA algorithms as well as data visualization.

"**Segmentation**" refers to the process of manually placing customers into multiple groups based on similarity. Demographic classification, and customer classification are applicable and customers are classified according to "Marketer-set criteria", so it might non-statistical methods. Therefore, simple customer segmentation relies on intuition, which can fail to produce desired results and get errors.

So this requires **"clustering."** This is an automated, statistically rigorous process of finding their similarities so that customers can be grouped. Also, this is a way to automatically discover segmentation that exist within the customer base using various elements of the customer by discovering who they target, rather than knowing who they are targeting in advance. Using K-means, one of representative clustering, we assume a separable spherical clustering so that the mean converges to a cluster-centered one.

RELATED WORK

There have been many attempts which segment to target customer segmentation. The process of dividing customers with the similar segment and different patterns into different segments is called customer segmentation. One study proposes a new customer segmentation model in which purchase scores are assigned according to the distribution, reflecting the purchased quantity and amount with the customers' needs by analyzing the customer's purchase history data. After grasping the characteristics of clustering formed using K-means, the quality of clustering is quantitatively evaluated and compared using silhouette scores.

RELATED WORK

Incorporating K-means, Hierarchical Clustering and PCA in Customer Segmentation

Azad Abdulhafedh*

University of Missouri, USA
*Corresponding author: asa8cd@mail.missouri.edu

Received January 04, 2021; Revised January 25, 2021; Accepted February 02, 2021

Abstract This paper addresses the use of clustering algorithms in the customer segmentation to define a marketing strategy of a credit card company. Customer segmentation divides customers into groups based on common characteristics, which is useful for banks, businesses, and companies to improve their products or service opportunities. The analysis explores the applications of the K-means, the Hierarchical clustering, and the Principal Component Analysis (PCA) in identifying the customer segments of a company based on their credit card transaction history. The dataset used in the project summarizes the usage behavior of 8950 active credit card holders in the last 6 months, and our aim is to perform customer segmentation in the most accurate way using clustering techniques. The project uses two approaches for customer segmentation: first, by considering all variables in the clustering algorithms using the Hierarchical clustering and the K-means. Second, by applying the dimensionality reduction through Principal Component Analysis (PCA) to the dataset, then identifying the optimal number of clusters, and repeating the clustering analysis with the updated number of clusters. Results show that the PCA can effectively be employed in the clustering process as a check tool for the K-means and Hierarchical clustering.

Keywords: *K-means, Hierarchical Clustering, Principal Component Analysis, Agglomerative hierarchical clustering, scree plot, Silhouette average width, Davies-Bouldin Index, Dunn index, customer segmentation*

Cite This Article: Azad Abdulhafedh, "Incorporating K-means, Hierarchical Clustering and PCA in Customer Segmentation." *Journal of City and Development*, vol. 3, no. 1 (2021): 12-30. doi: 10.12691/jcd-3-1-3.

1. Introduction

Unsupervised learning is a process for gaining meaningful insights by summarizing data in innovative ways. As opposed to supervised learning methods that predict a target of interest; in an unsupervised learning, no single feature is more important than any other. Thus, with unsupervised learning, there are inputs but no supervising output [1,2,3]. Customer Segmentation (also called market segmentation) is one the most important applications of the unsupervised learning methods in data science and machine learning. Customer Segmentation is the process of dividing customers into several groups that share common characteristics relevant to marketing such as gender, age, interests, and miscellaneous spending habits. Segmentation process can help businesses and companies understand their customer groups, target the right groups, and develop effective marketing strategies for different targeted groups. Clustering techniques are the most appropriate methods that enable businesses and companies to identify segments or groups of customers in order to target the potential user base. Customer segmentation can be performed using a variety of different customer characteristics. The most common types are customer's geographical regions, customer's demographics (e.g., age, gender, marital status, income), customer's psychographics (e.g., values, interests, lifestyle, group affiliations), and purchase behavior (e.g., previous purchases, shipping preferences). Customers data usually contain observations from millions of customers; however, these customers may only belong to a few segments: customers are similar within each segment but different across segments. Grouping data with similar characteristics into clusters is called cluster analysis. This is similar to classification; except we do not have a labelled dataset to use for training. Data points are simply grouped based on how similar they are to each other. Since there is not a response variable, this is an unsupervised method, which implies that it seeks to find relationships between the observations without being trained by a response variable [1,4,5,6].

2. Data

A credit card company has collected over the time data about their customers' accounts. The data has various facts related to the customers, such as their balances, purchases, cash advances, credit scores, etc. The management team was willing to make meaningful insights from the data, and then develop strategies to target segments of customers in order to increase credit card sales, and in turn to increase the revenue. The dataset

---

Journal of Applied Data Sciences
Vol. 2, No. 1, January 2021, pp. 19-25

ISSN 2723-6471

Maximizing Strategy Improvement in Mall Customer Segmentation using K-means Clustering

Musthofa Galih Pradana [1,*], Hoang Thi Ha [2],

[1] Department of Informatics Alma Ata University, Yogyakarta, Indonesia
[2] Department Management Information System University of Danang, Vietnam
[1] mgalihpradana@almaata.ac.id; [2] hoang2th@due.udn.vn
* corresponding author

(Received December 21, 2020 Revised January 4, 2021, Accepted January 14, 2021, Available online January 15, 2021)

Abstract

The application of customer segmentation is very vital in the world of marketing, a manager in determining a marketing strategy, knowing the target customer is a must, otherwise it will potentially waste resources to pursue the wrong target. Customer segmentation aims to create a relationship with the most profitable customers by designing the most appropriate marketing strategy. Many statistical techniques have been applied to segment the market but very large data are very influential in reducing their effectiveness. The aim of clustering is to optimize the experimental similarity within the cluster and to maximize the dissimilarity in between clusters. In this study, we use K-means clustering as the basis for the segmentation that will be carried out, and of course, there are additional models that will be used to support the research results. As a result, we have succeeded in dividing the customer into 5 clusters based on the relationship between annual income and their spending score, and it has been concluded that customers who have high-income levels & have a high spending score are also very appropriate targets for implementing market strategies.

Keywords: Segmentation, Strategy, Clustering, K-Means.

1. Introduction

In this era, increasing the level of consumer consumption is very reasonable, this is based on the very fast development of production. This makes each person feel like they have an obligation to spend something to enjoy these developments. At this point, an increase in the number and variance of products is not a bad thing for the market, but an increase in customers can sometimes lead to wasted resources due to a strategy aimed at the wrong customer [1]. At this time a lot of managers and people who work in the marketing field try various things to create the right market strategy. However, we are talking about customers who are human and change or can change based on various factors. Many applications of certain strategies such as discounts, annual promotions, memberships, etc. may work for a while but after that, it is nothing more than a waste of resources, both energy, and money.

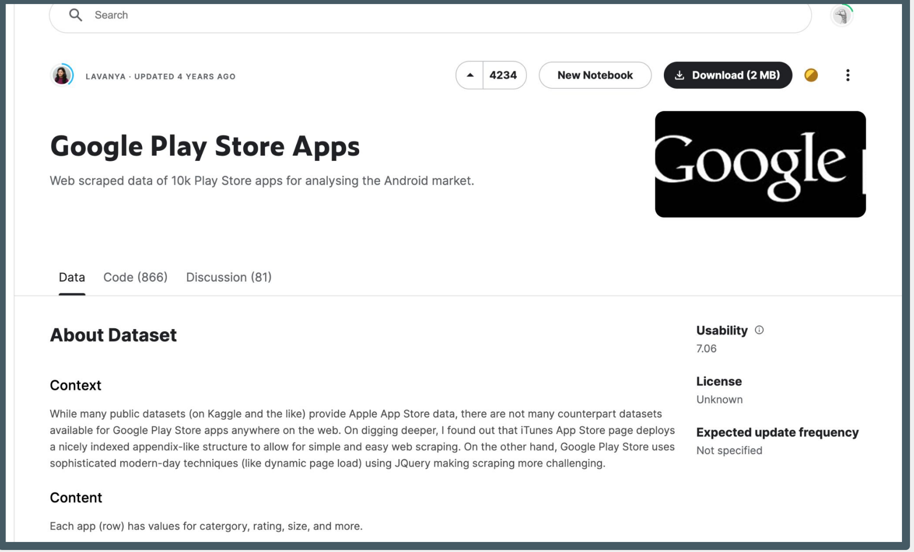
As a manager, it is very important to be able to recognize the patterns or habits of the customers themselves. As a matter of fact, the mall Industry is often involved in a race to increase their customers and therefore make huge profits [2]. There are several factors that explain why the mall rejects its role. First, the level of customer activity is higher, they have less time to shop, and finally, they reduce their shopping frequency. In fact, there are too many of the same malls in a district or city and eventually, customers will go to the shopping center that offers the most products and the best service. This factor encourages mall managers to develop a strategy to differentiate them from competitors [3, 4].

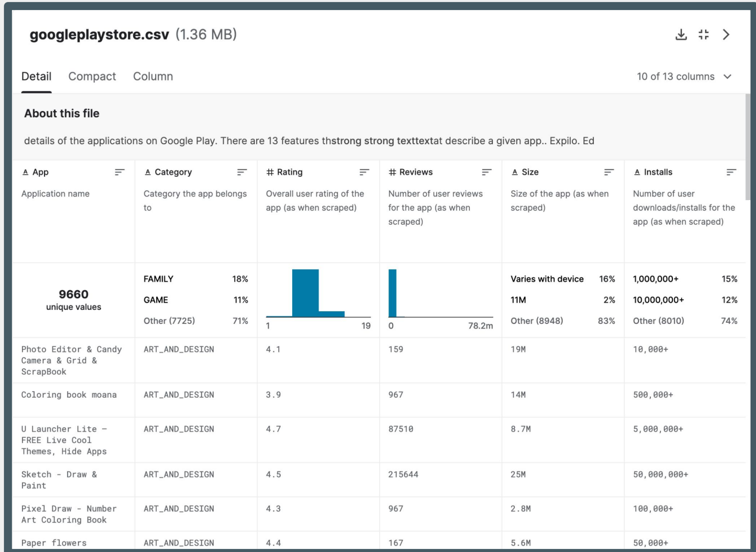
2. Literature Review

research shows that detecting where a customer is going to meet their shopping needs is highly dependent on the service from the provider and the characteristics of the place they are going to. In certain perspectives such

OUR DATASET:

https://www.kaggle.com/datasets/lava18/google-play-store-apps





OUR GOAL AND SIGNIFICANCE OF THE PROJECT

Hypothesis: Certain application categories prioritize their app ratings over number of installs, or the other way around.

This can determine the app's primary goal and their target audience. i.e. Automobile apps prioritize ratings over number of installs because they want to adhere to a smaller target audience who will only download the state-of-the-art app, VERSUS Social Media apps who just want the most amount of installs because their target audience is their priority.

METHODS:

01 CLEANING DATA

We began with cleaning up our data, keeping the parts related to our project.

02 DATA VISUALIZATION

We did exploratory data visualization to better understand our data and run k means.

03 K-MEANS

We ran K-means to separate our data to K clusters and observe patterns,

04 PCA

Next we ran PCA to reanalyze our data with more information.

CLEANING DATA

For our purposes, comparing installs and ratings, we do not need to use all of the data to run K-means or PCA. Therefore we began with dropping the size, type, genres, last updated, current version and android version columns.

```
[ ] # We care about all columns besides the Size, Type (Same as Price), Genres (Same as Category), Last Updated, Current Version, Android Version.
    df = df.drop(['Size', 'Type', 'Genres', 'Last Updated', 'Current Ver', 'Android Ver'], axis = 1)
    df.head()
```

	App	Category	Rating	Reviews	Installs	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	10,000+	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	500,000+	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	5,000,000+	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	50,000,000+	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	100,000+	0	Everyone

DATA VISUALIZATION

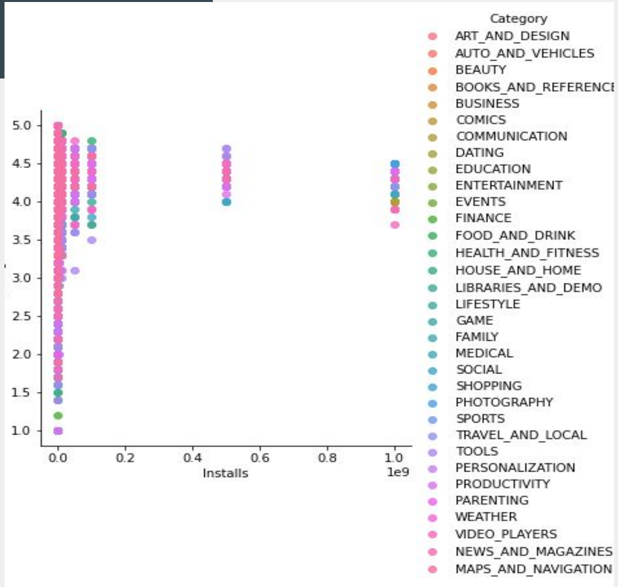
Our next step was to to do some exploratory data visualization, we created some scatter plots on Installs vs Ratings coordinated by category.

```python
[ ] # Let's filter out only the rows that have a valid rating <= 5
    df = df[df['Rating'] <= 5]

    # Change Installs from strings to integers
    df['Installs'] = df['Installs'].str.replace(r'\D', '').astype(int)

    # Only want to work with apps with 100+ installs
    df = df[df['Installs'] >= 100]
    df.head()
```

```
<ipython-input-5-743675eda545>:5: FutureWarning: The default value of regex will change from True to False in a future version.
  df['Installs'] = df['Installs'].str.replace(r'\D', '').astype(int)
```

	App	Category	Rating	Reviews	Installs	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	10000	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	500000	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	5000000	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	50000000	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	100000	0	Everyone

```python
# Scatter plot on Installs vs Ratings, color coordinated by category
sns.lmplot('Installs', 'Rating', data=df, hue='Category', fit_reg=False)
```



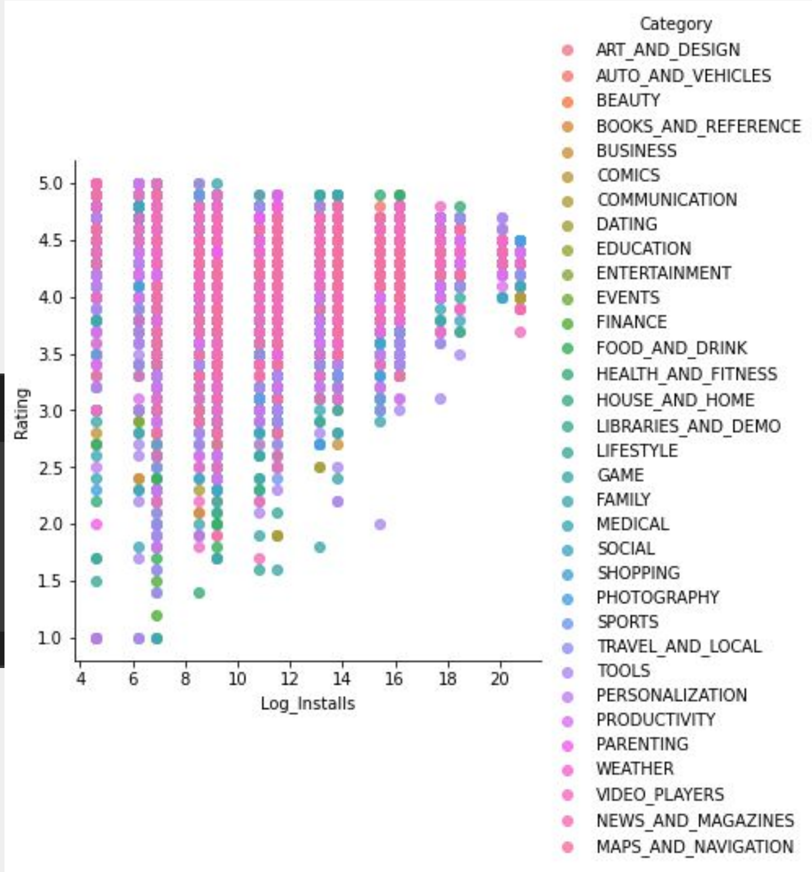
We saw that the ranges for installs for too sparse out causing our scatterplot to be too spread out, we fixed this issue by taking the log of install column and then randomizing the numbers from its lower boundary to the next boundary up.

Representation on log scale:

```
[ ] # 1. Represent on a log scale
    df['Log_Installs'] = np.log(df['Installs'])
    df.head()
```

	App	Category	Rating	Reviews	Installs	Price	Content Rating	Log_Installs
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	10000	0	Everyone	9.210340
1	Coloring book moana	ART_AND_DESIGN	3.9	967	500000	0	Everyone	13.122363
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	5000000	0	Everyone	15.424948
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	50000000	0	Teen	17.727534
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	100000	0	Everyone	11.512925

```
[ ] sns.lmplot('Log_Installs', 'Rating', data=df, hue='Category', fit_reg=False)
```

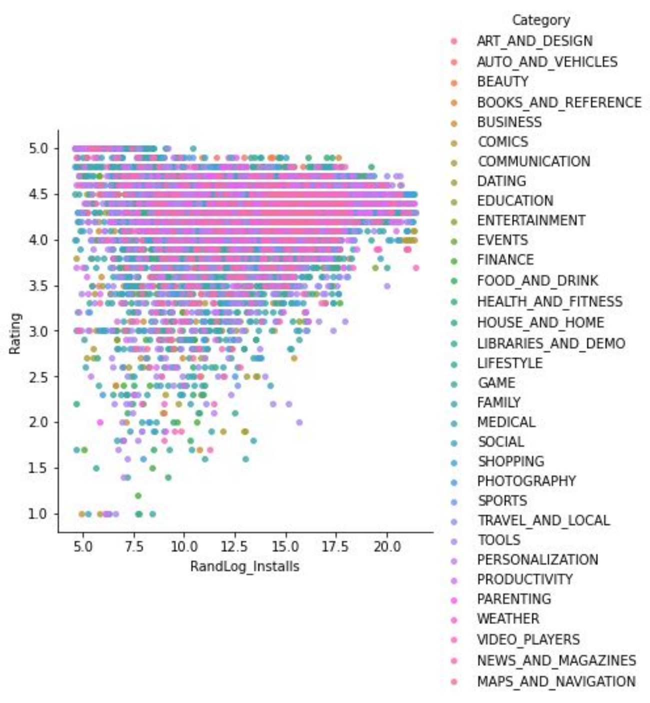


Randomizing install numbers:

```
[ ] # 2. Randomize install numbers between the ranges
    log_ranges = sorted(set(df['Log_Installs'].to_numpy()))
    print(log_ranges)

    def rfunc(x):
      # preset stopping point in case x ends up being last number in log_ranges
      # ln(1,000,000,000) = 20.72326583694641
      # make stopping point ln(2,000,000,000) = 21.41641302
      # because 5 billion is pretty unrealistic
      stop = 21.41641302

      for i in range(len(log_ranges)-1):
        if x == log_ranges[i]:
          stop = log_ranges[i+1] - .000000000000001    # bc random.uniform is inclusive
      return random.uniform(x,stop)

    df['RandLog_Installs'] = df['Log_Installs'].apply(lambda x: rfunc(x))
    df = df.drop('Log_Installs', axis=1)
    df.head()
```

```
[4.605170185988092, 6.214608098422191, 6.907755278982137, 8.517193191416238, 9.210340371976184, 10.819778284410283, 11.512925464970
```

	App	Category	Rating	Reviews	Installs	Price	Content Rating	RandLog_Installs
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	10000	0	Everyone	10.448626
1	Coloring book moana	ART_AND_DESIGN	3.9	967	500000	0	Everyone	13.647642
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	5000000	0	Everyone	15.920619
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	50000000	0	Teen	18.339740
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	100000	0	Everyone	12.991252

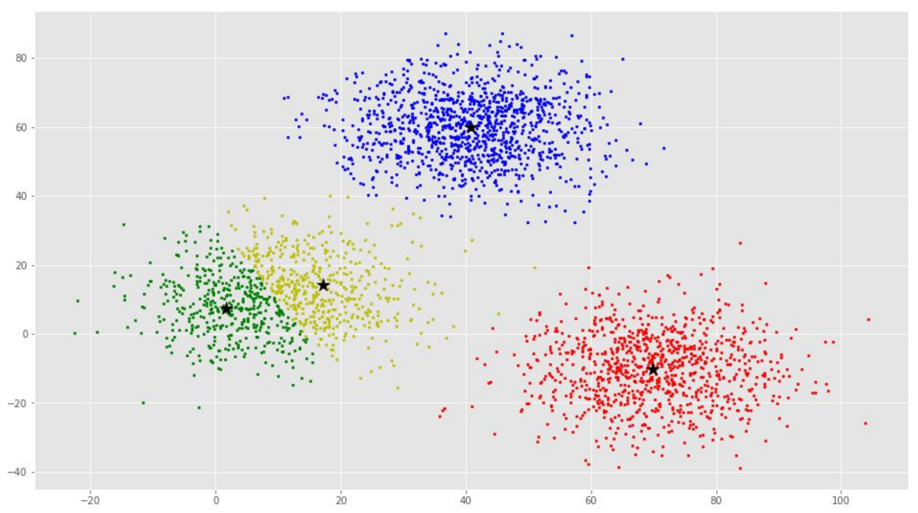


K-MEANS

K-means is an iterative batch algorithm that allows us to partition our data into K disjoint clusters. In the case of our data, it allows us to cluster based on app installs and ratings.

K-MEANS

Our goal: To partition the data into K clusters, which can possibly tell us how the clusters differ from each other. By reapplying app category labels after the dataset has been clustered, we hope to see some underlying pattern between the categories.

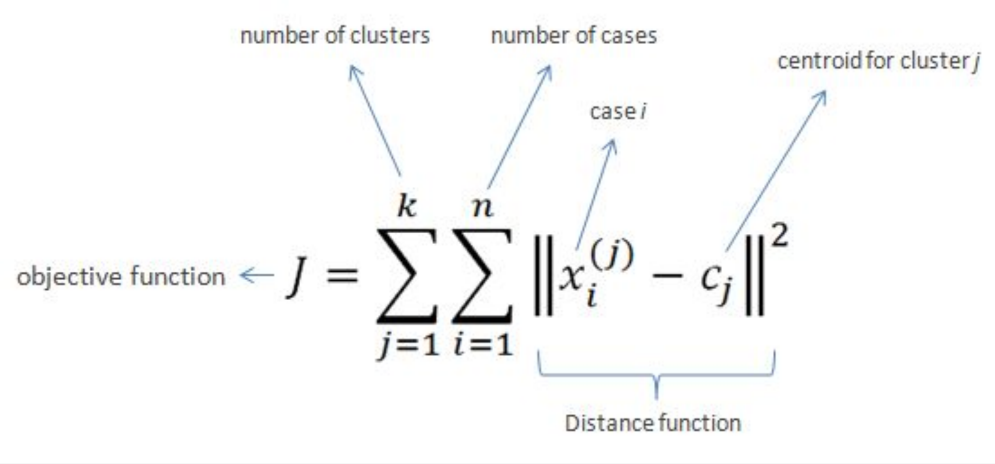


K-MEANS

Step 1: Determine our desired number of clusters

- We are coming in with the expectation that apps will be divided into 4 categories (hypothesis):
  - high ratings high installs
  - low ratings high installs
  - high ratings low installs
  - low ratings low installs
- With that in mind, we will begin with K = 4.
  - We expect clustering like the image above.

K-MEANS



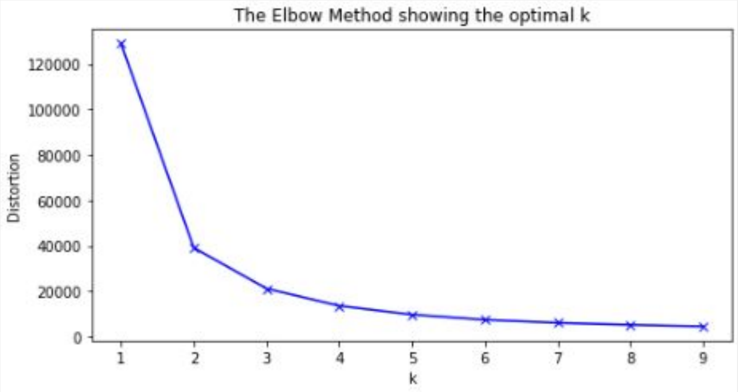
$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$$

Step 2: Select random points to be cluster centers (4 random points), and iteratively refine clusters using K-means algorithm

- Assign every point to one of the four clusters
- Recalculate each cluster center to be the average of all the points assigned to it
- Reassign points to clusters
- Repeat until convergence

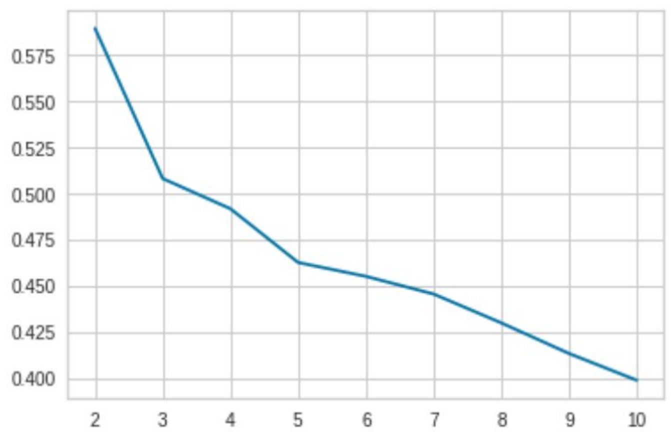
To simplify, we are minimizing the distance between each point and its cluster center iteratively by adjusting the cluster center.

K-MEANS



Step 3: Determine the best value of K given our data

- We used two methods
- The elbow method which gives us a measure of the distortions/variances of our k-means. The point of inflection on the curve (elbow) is our ideal K value
- The silhouette score method which gives us a metric for how well our data is clustered and how different each cluster is from others



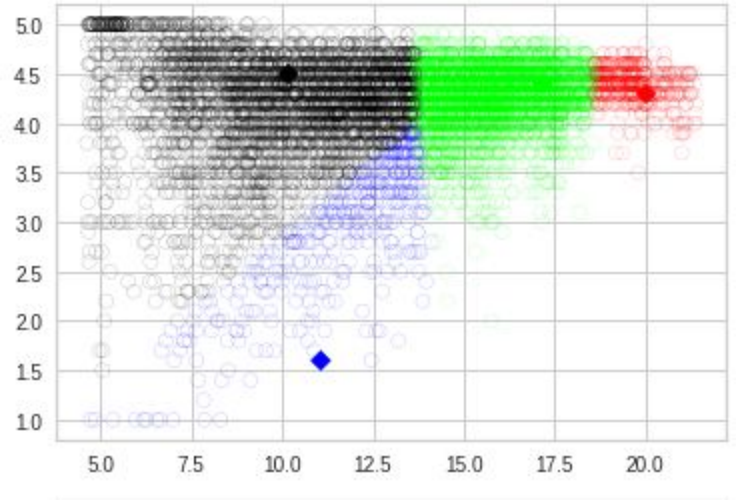
K-MEANS

HERE IS OUR CODE

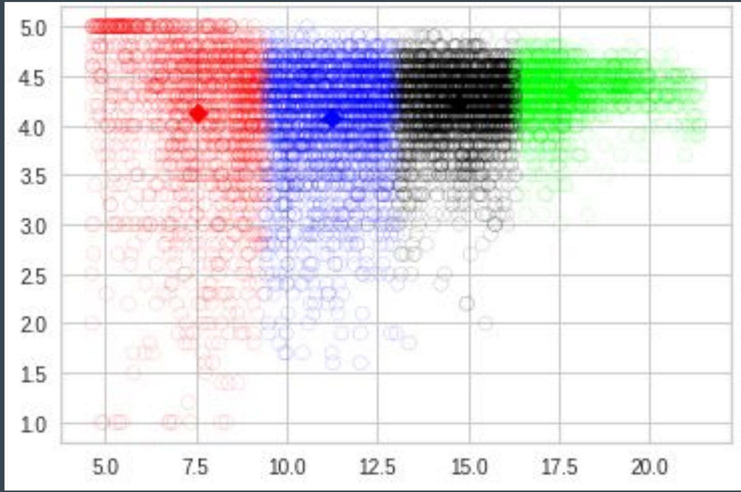
```python
def calcSqDistances(X, Kmus):
    # We will be calculating the eucladian distance between the points
    # Calculate matrix product
    product = X @ Kmus.T

    # Square then sum all values in each matrix
    Xsquared = np.square(X)
    Ksquared = np.square(Kmus)
    Xsum = np.sum(Xsquared, axis=1)
    Ksum = np.sum(Ksquared, axis=1)

    # multiply the product matrix by -2 first before adding to the others
    product *= -2
    sqDmat = (product + Ksum.T).T + Xsum

    return sqDmat.T

def recalcMus(X, Rank):
    # Recalculate the center of the cluster
    # np.divide - matrix divide, to take average divide all of them
    sumranks = np.sum(Rank, axis=0)
    product = X.T @ Rank
    # divide by the sum to get averages and then retranspose
    avgs = np.divide(product, sumranks).T
    return avgs

def runKM(K, df):
    # Allocate space for Kmus
    Kmus = np.zeros((K, df.shape[1]))


    # Randomly picking points from the data as cluster centers
    rndinds = np.random.permutation(df.shape[0])
    Kmus = df[rndinds[:K]];

    # Maximum iterations to run k means algorithm
    maxiters = 1000;

    for iter in range(maxiters):
        # Calculate squared distances between data and mu vectors. Given the
        # minimum values we recalculate our mu vecotors

        # sqDmat will be an N-by-K matrix with the squared distance from the nth
        # data vector to the kth mu vector
        sqDmat = calcSqDistances(df, Kmus);

        # given the matrix of squared distances, determine the closest cluster
        # center for each data vector

        # Get number of columns
        cols = len(sqDmat[0])
        # Create matrix with 1's across the diagonals and zeros everywhere else
        diagonals = np.eye(cols)
        # Get the minimum values in sqDmat
        minvals = np.argmin(sqDmat, axis=1)
        # Set rank equal to the matrix showing which columns hold the smaller values
        Rank = diagonals[minvals]

        # Save current Kmus to KmusOld
        KmusOld = Kmus
        plotCurrent(df, Rank, Kmus)
        plt.show()

        # Recalculate kmus/center of clusters
        Kmus = recalcMus(df, Rank)

        # K means is done when cluster centers have converged
        if sum(abs(KmusOld.flatten() - Kmus.flatten())) < 1e-6:
            break
    plotCurrent(df,Rank,Kmus)
```

K-MEANS

Initial Clusters

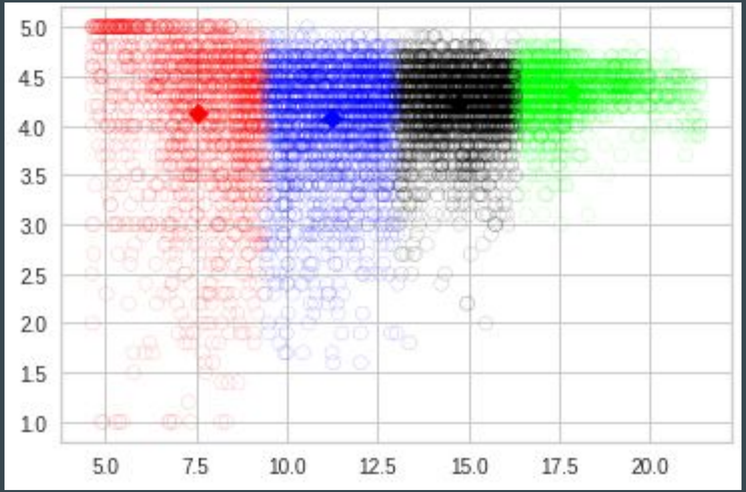


Final Clusters

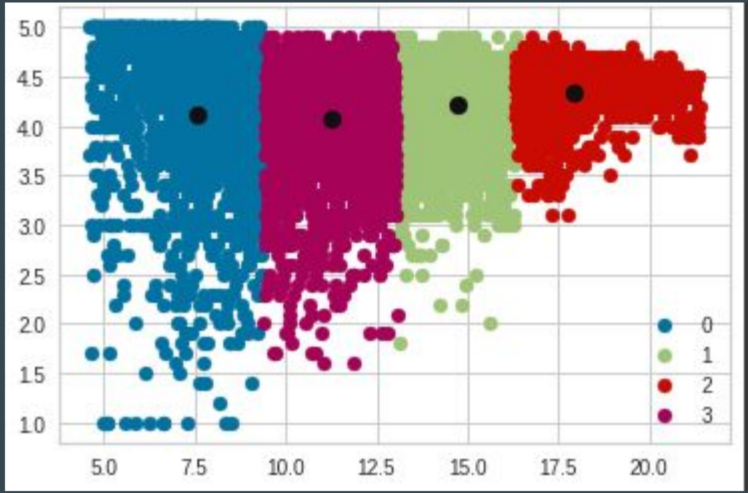


K-MEANS

Comparison with Sklearn's KMeans function



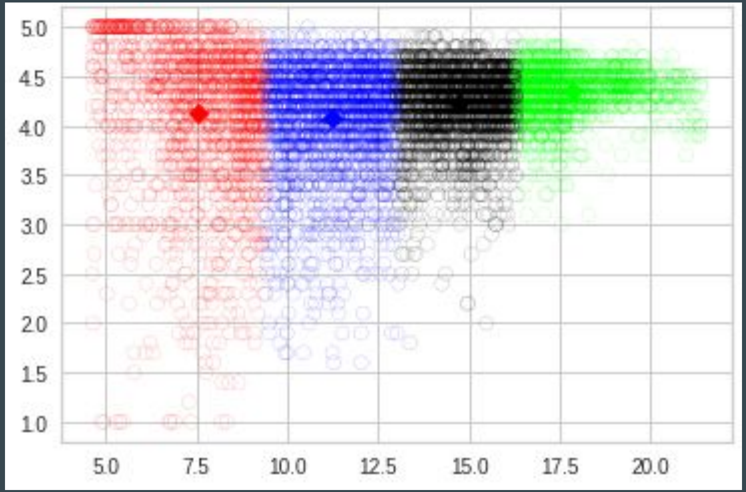
Original



Sklearn

K-MEANS RESULTS

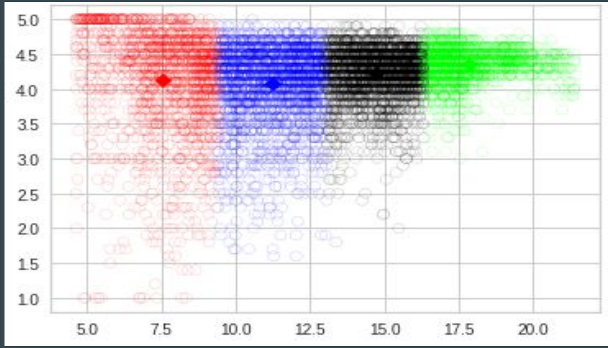
- The resulting clusters were different from our expectations. It appears that these clusters are divided into columns
- This is because ultimately, the largest square distance occurs between the Install ranges. So since it's clustering groups by smallest square distance, it's going to group them vertically and appear as columns. This could be of 2 reasons:
    - 1. Our data does not vary enough in Ratings vs Installs. Meaning there are no App Categories that significantly favor Ratings over Installs, or the other way around.
    - 2. Our Ratings (floats between 1 and 5) are too dense, so there will be no clear division between Ratings for k-means to split along.



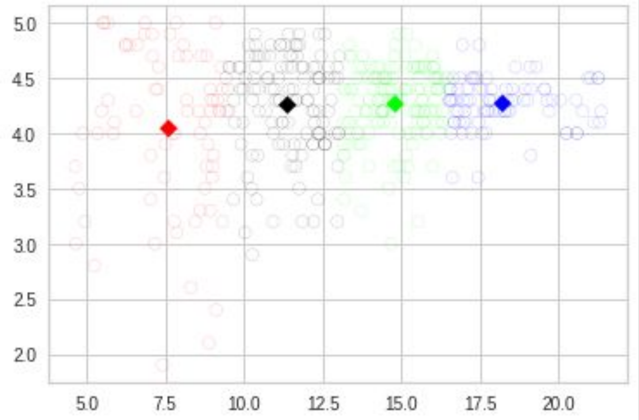

```
df['Installs'].unique()

array([      10000,      500000,     5000000,    50000000,      100000,
             50000,     1000000,    10000000,        5000,   100000000,
        1000000000,        1000,   500000000,         100,         500])
```

K-MEANS RESULTS

- We tried to account for different app categories potentially having an effect on the distribution of our data
    - For example, we attempt to filter our categories from 31 unique values to 4. However, this yields clusters that have a similar shape to the ones before.
- K-means would have better supported our data, if for example, we had data that clearly bunched up closer to 4 separate sections of the grid plane: low rating low installs, low rating high installs, high rating low installs, high rating high installs. but in this case, our data evenly spreads across the entire grid with clear divisions between the Installs range



Original data with all categories included

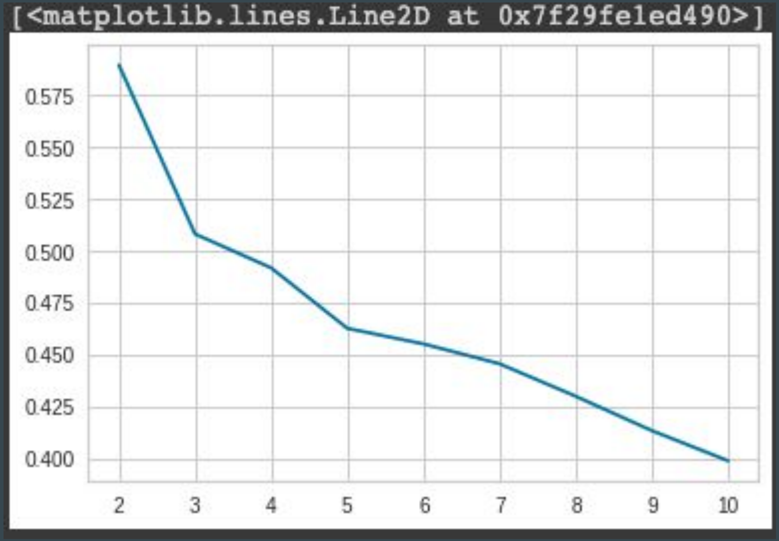


Data with only beauty, comics, social, and auto/vehicle categories

K-MEANS RESULTS

- Silhouette Score:
  - Evaluates the quality of clusters in K-Means.
  - Uses mean distance within clusters and mean distance between clusters
    - Essentially how similar a point is to its own cluster versus other clusters
  - [-1, 1] Score of 1 indicates cluster is dense and well separated from other clusters, while 0 represents overlapping clusters. A negative score can indicate samples were assigned to wrong clusters.
  - With K=4, the clustering received a score of **0.492.**
- Optimal Value of K appears to be K = 2, with a score of **0.589**

$$s = \frac{b-a}{\max(a,b)}$$

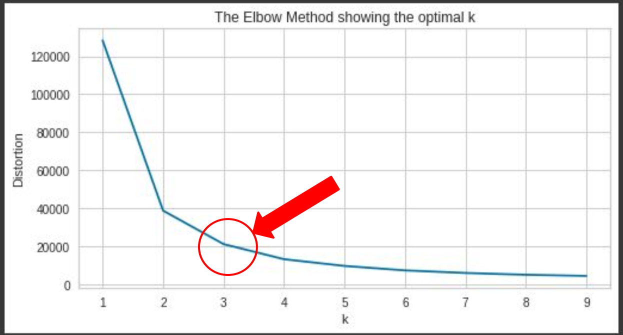


K-MEANS RESULTS

- Elbow method:
  - Calculate sum of squared error for each value of K
  - Plot the SSE vs K on graph and look for “elbow”
    - Indicates point of diminishing SSE for increasing value of K.
- Plot indicates optimal value of K is 3.

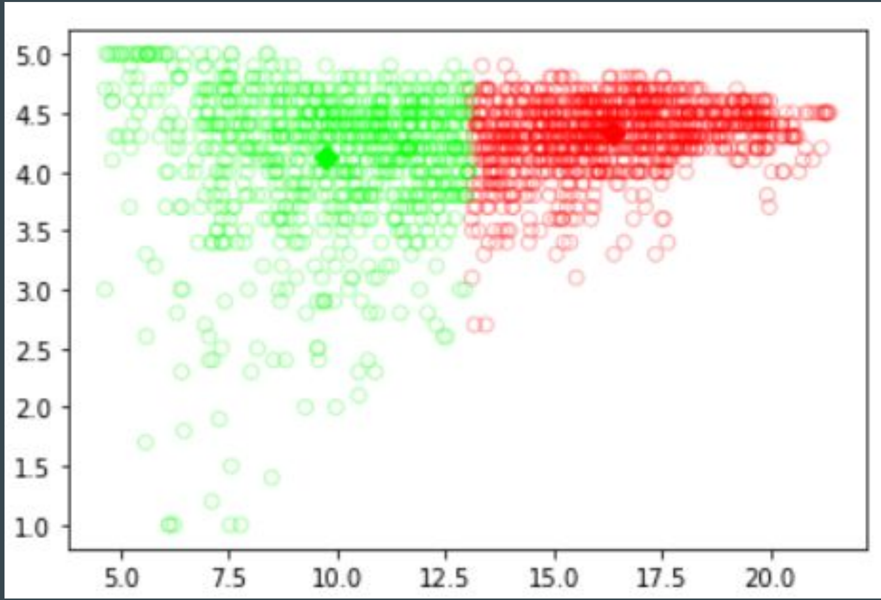
  Which was around K = 4.

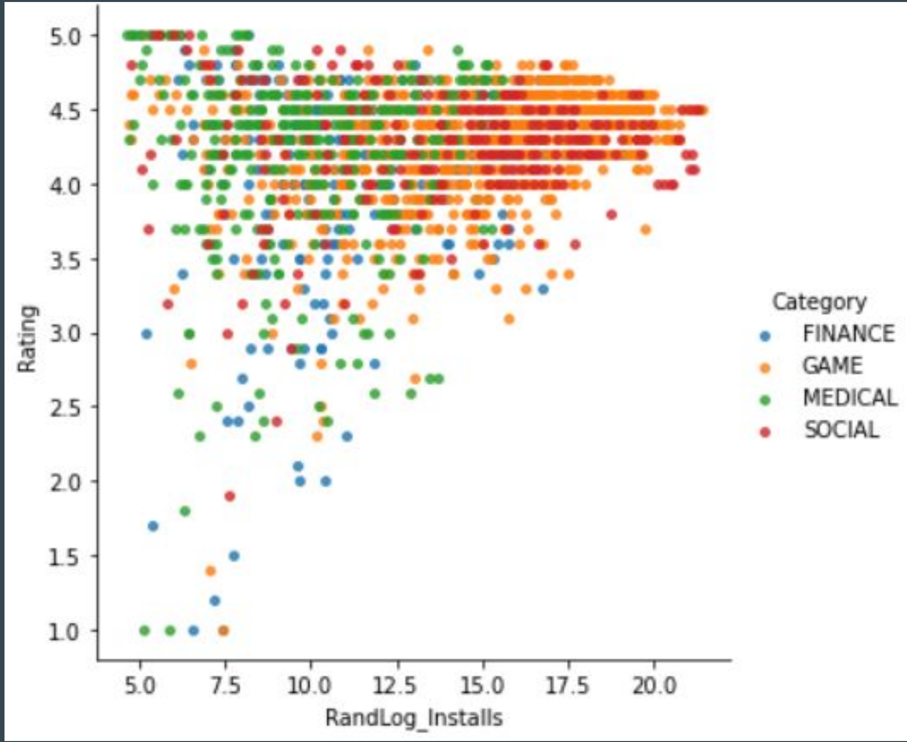
```
[178] distortions = []
      K = range(1,10)
      for k in K:
          kmeanModel = KMeans(n_clusters=k)
          kmeanModel.fit(df[["RandLog_Installs", "Rating"]])
          distortions.append(kmeanModel.inertia_)

      plt.figure(figsize=(8,4))
      plt.plot(K, distortions, 'bx-')
      plt.xlabel('k')
      plt.ylabel('Distortion')
      plt.title('The Elbow Method showing the optimal k')
      plt.show()
```



HYPOTHESIS FINDINGS

- By cutting down the amount of Categories we have, we could better compare the Apps and see which Rating vs Installs groups they fall into.





PRINCIPAL COMPONENT ANALYSIS

In hopes of obtaining more distinct clustering of our Apps, we will use PCA to take advantage of the rest of our available data. So by utilizing our Reviews and Content Rating columns, we could derive principle components that better explain which Apps are similar to each other.

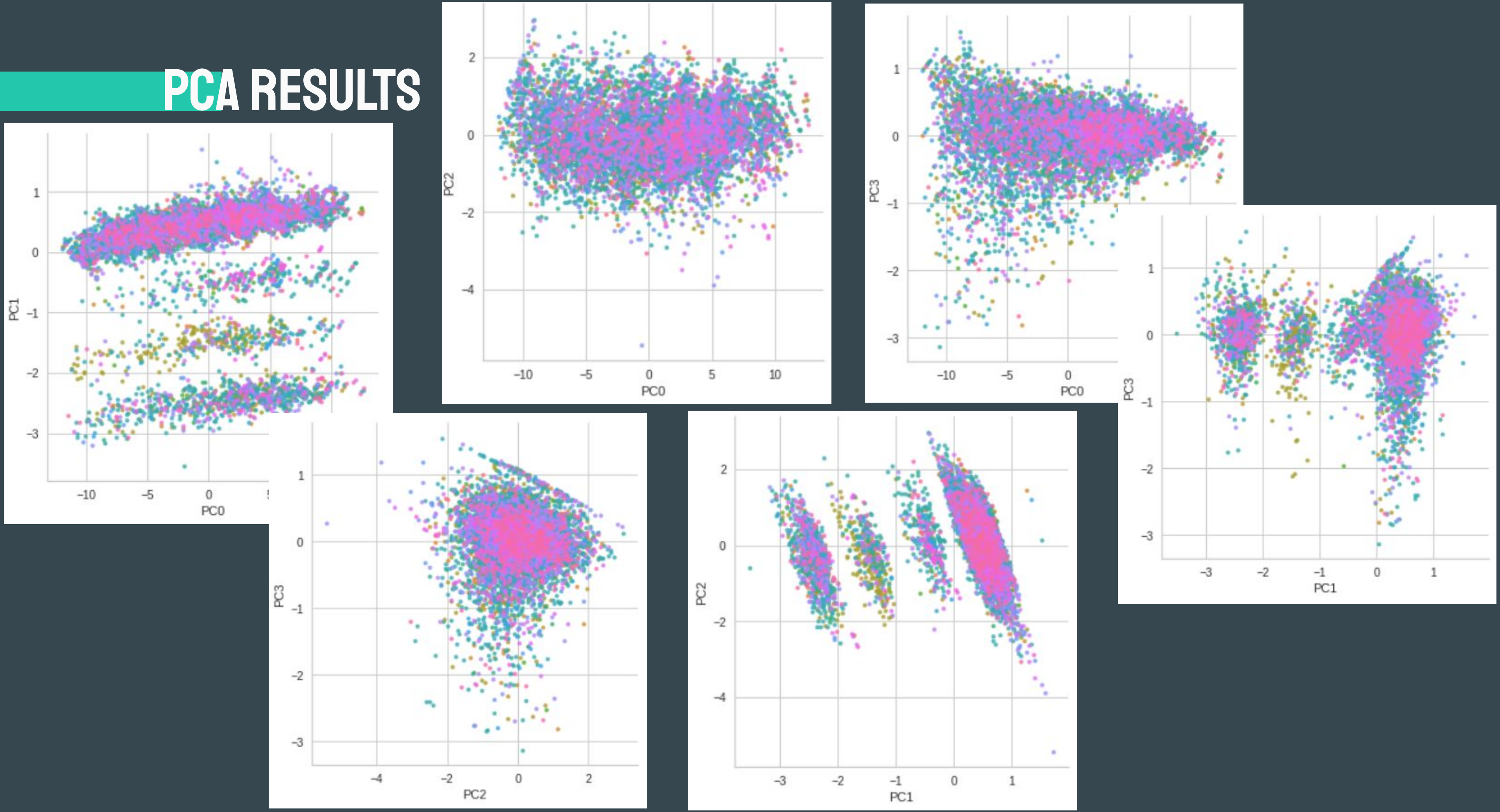
```
[23] # Use LabelEncoder to turn Content Rating into numerical representations
     le = preprocessing.LabelEncoder()
     le.fit(df2['Content Rating'])
     list(le.classes_)
     df2['Content Rating'] = le.transform(df2['Content Rating'])
```

```
[24] # Change Installs from strings to integers
     df2['Reviews'] = df2['Reviews'].str.replace(r'\D', '').astype(int)
     # Log the reviews section so the range isn't crazy either
     df2['Reviews'] = np.log(df2['Reviews'])
     df2.head()
```

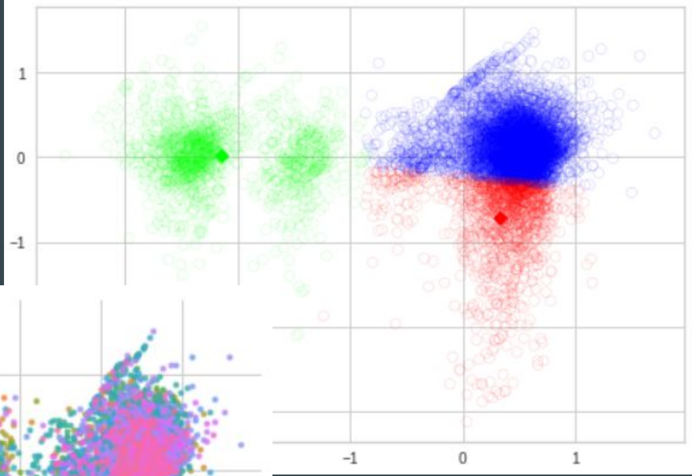
```
<ipython-input-24-f40a43d1515b>:2: FutureWarning: The default value of regex
  df2['Reviews'] = df2['Reviews'].str.replace(r'\D', '').astype(int)
```

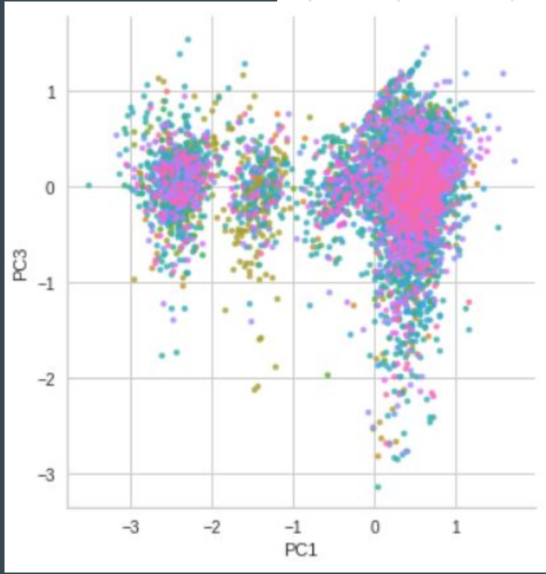
	Rating	Reviews	Content Rating	RandLog_Installs
0	4.1	5.068904	1	9.506147
1	3.9	6.874198	1	13.251732
2	4.7	11.379508	1	16.071323
3	4.5	12.281384	4	17.854347
4	4.3	6.874198	1	12.388603

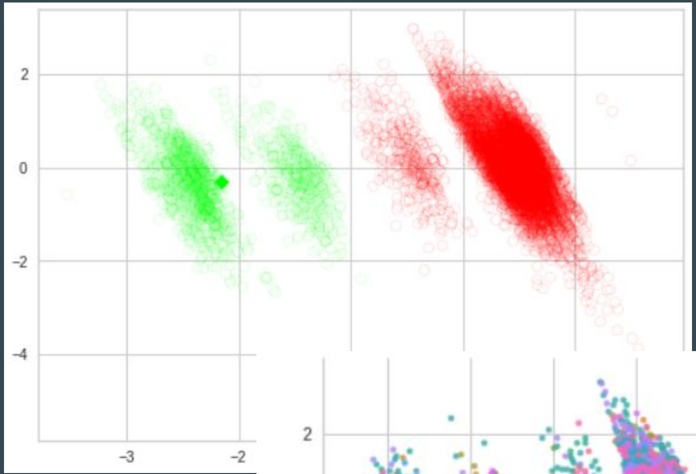
PCA RESULTS

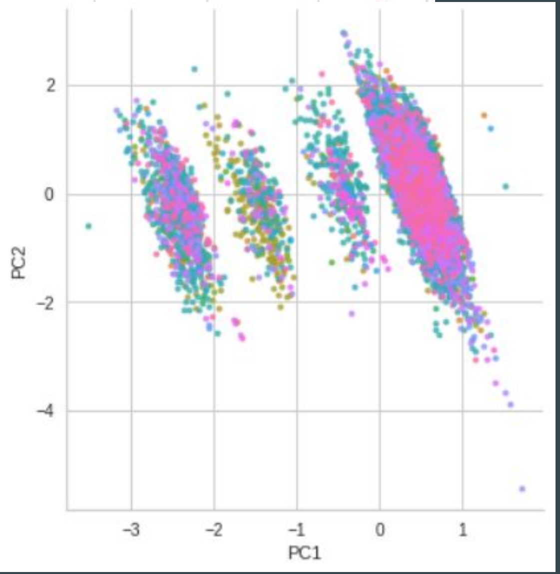


PCA + K-MEANS RESULTS









As we can see here, running K-means on our principal components is more successful in clustering the applications than K-means alone.

DISCUSSION, WHAT WE LEARNED:

Our normal K-Means on the Ratings vs Installs data would better support our exact hypothesis, since we are able to pinpoint which Apps are higher and lower on the Ratings and Installs spectrums. However, PCA does far better with grouping similar Apps with each other by using the rest of the available data in our dataset.

Looking further in, comparing Rating vs Install numbers was not sufficient enough to cluster the Apps into groups of similar neighbors, and therefore we still were not able to see which Apps favor Ratings over Installs. However, with PCA, we could now see the colored Categories falling into tighter groups with each other, and could therefore derive a new Hypothesis of which Google Play Store Apps are more similar to each other in terms of Ratings, Installs, Review Numbers, and Content Rating.

Possible Extensions/Improvements:

- One potential improvement is finding a better dataset
  - Our dataset was very clustered and more or less uniformly distributed
    - This made it difficult to get a good result from K-means
  - Would have been better if there were already visible clusters in the data
  - To solve it this problem in our project we did clusters on subgroups in the data
- One potential extension is testing our model on new data
  - Get uncategorized data and use that on our model
  - Our model will predict what category the app is from ie beauty, game, or medical
  - Measure the accuracy of our model and potentially readjust it

CITATIONS:

- Data set : https://www.kaggle.com/datasets/lava18/google-play-store-apps
- Lectures: COGS 188B

- Abdulhafedh, Azad. "Incorporating k-means, hierarchical clustering and pca in customer segmentation." *Journal of City and Development* 3.1 (2021): 12-30.
- Pradana, Musthofa. "Maximizing Strategy Improvement in Mall Customer Segmentation Using K-Means Clustering." *Journal of Applied Data Sciences*, vol. 2, no. 1, 2021, doi:10.47738/jads.v2i1.18.

THANKS!!!