**Module 6**

* [Video Transcripts](https://student.emeritus.org/courses/4765/files/3055193?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/3055193/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/3055194?wrap=1)

**Notes:**

Principal Component Analysis (PCA)

Singular Value Decomposition (SVD)

KMeans and DBScan clustering

Density-based spatial clustering of applications with noise (DBSCAN)

**DBSCAN**

A density-based clustering non-parametric algorithm that groups together data points that are closely packed together, marking as outliers points that lie alone in low-density regions

**K-Means**

A method of vector quantization that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster

**K-Means++**

An algorithm for choosing the initial values or ‘seeds’ for the k-means clustering algorithm

**Principal Component Analysis (PCA)**

The process of computing principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest

**Singular Value Decomposition (SVD)**

The factorization of a real or complex matrix

mu = X.mean()

sigma = X.std()

Xnorm = (X - mu)/sigma

Sigma = np.diag(sigma)

U, Sigma, VT = svd(Xnorm)

reconstruct\_X = U @ Sigma @ VT

percent\_variance\_explained = sigma / sigma.sum()

print(np.cumsum(percent\_variance\_explained[:22]))

np.allclose(df['cluster label0'], df['cluster label1'])

**Module Issues:**

**Activity 6.1 Problem 5:** “Standardize the singular values in the first ten entries of Sigma by dividing each by the sum of the main diagonal”, the statement is not clear to use sum of diagonal of Sigma!

**Activity 6.2 Problem 4:** Solution does not normalize dataset before SVD which is wrong!



**Activity 6.6 Problem 2:** hue = 'label' is supposed to be 'cluster label’!

**Activity 6.7 Problem 3:** variable inertia\_8\_centers is supposed to be inertia

**Try-It activity 6.1:**

credit.csv is missing in data folder! Fixed.

'default payment next month' supposed to be 'default.payment.next.month'

**Try-It activity 6.2:**

images/segments.jpeg is missing in the zip file.

**Quizes:**

PCA is used for clustering. :- False

*You are correct! The answer “False” is correct because the principal component analysis is used for dimensionality reduction.*

PCA looks for linear combinations of existing (blank) that capture the bulk of the variance. : Columns

*You are correct! The answer “Columns” is correct because PCA looks for new columns that are linear combinations of existing columns and capture the bulk of the variation in the data*.

The “curse of dimensionality” states that the amount of data you need to train a model increases exponentially with the number of inputs. : True

*You are correct! The answer “True” is correct because the amount of data you need to train a model increases exponentially with the number of inputs is stated as the “curse of dimensionality”.*

What does running SVD on X decompose X into? : U Σ V

*You are correct! The answer “*U Σ V*” is correct because running singular value decomposition (SVD) on X will decompose it into three matrices: U,*Σ,*and V.*

What is the formula to normalize a dataset X? : Xnorm = (X−μ)/σ

*You are correct! The answer “*Xnorm = (X−μ)/σ*” is correct because this is the formula to normalize the dataset X.*

What is the function in Python library “scipy.linalg” to compute the singular value decomposition? : svd()

*You are correct! The answer “*svd()*” is correct because this is the function in Python library “scipy.linalg” to compute the singular value decomposition.*

The Python function “numpy.allclose()” is used to find whether two arrays are element-wise equal. : True

*You are correct! The answer “True” is correct because the function “*numpy.allclose()*” is used to check whether two arrays in numpy are element-wise equal or not.*

In SVD, the matrix sigma has all the diagonal values as zero. : False

*You are correct! The answer “False” is correct because the matrix sigma in SVD is a diagonal matrix in which, other than the diagonal values, all other values are zero.*

Below is the equation to represent the multiplication of the matrices for SVD: ∑Di=1 σiuivit What does the parameter *D*represent? : Principal components

*You are correct! The answer “Principal components” is correct because in the formula for the multiplication of matrices in SVD, the parameter D represents the number of iterable principal components.*

The Matrix Σ in SVD has values “σi” in the diagonal of the matrix, which represent the importance of the i’th component for the dataset. : True

*You are correct! The answer “True” is correct because the value of*“σi”*conveys the importance of the i’th principal component for the dataset.*

The formula used to project the data into desired dimensions is

“ x̄rr  = Ur ∑r ”

The parameter “r” is defined as the total number of columns. : False

*You are correct! The answer “False” is correct because the parameter “r” is defined as the number of principal components that are selected.*

What is the symbol used for matrix multiplications? : @

*You are correct! The answer “@” is correct because this symbol is used for matrix multiplications.*

Clustering is a method for creating groups out of the columns of a dataset. : False

*You are correct! The answer “False” is correct because clustering is a method for creating groups out of the rows of a dataset.*

Clustering is an unsupervised machine learning model. : True

*You are correct! The answer “True” is correct because clustering has no labeled datasets.*

How is the centroid of a cluster “k” in k-means clustering represented? : μk

*You are correct! The answer “*μk*” is correct because the mean of each cluster is declared as the centroid of that cluster, which, for cluster k, is “*μk*”.*

In the K-means clustering algorithm, how is inertia defined? : Sum of the squared distances from points to their centroids

*You are correct! The answer “Sum of the squared distances from points to their centroids” is correct because inertia in K-means is the summation of the squared distances of data points from their respective centroids.*

The stepwise sequence for the K-means algorithm is as follows:

1. Assignment
2. Updation

: False

*You are correct! The answer “False” is correct because K-means starts with updating of centroids initially and then assigning data points to the nearest centroids.*

What is the stopping criteria for K-means clustering? : Assignment step has no change of data points

*You are correct! The answer “*Assignment step has no change of data points*” is correct because the stopping criteria for K-means clustering is when assignment stops changing.*

Consider this dataframe:

Dataframe

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CustomerID** | **Gender** | **Age** | **Annual Income (k$)** | **Spending Score (1-100)** |
| 1 | Male | 19 | 15 | 39 |
| 2 | Male | 21 | 15 | 81 |
| 3 | Female | 20 | 16 | 6 |
| 4 | Female | 23 | 16 | 77 |
| 5 | Female | 31 | 17 | 40 |
| ... | ... | ... | ... | ... |
| 196 | Female | 35 | 120 | 79 |
| 197 | Female | 45 | 126 | 28 |
| 198 | Male | 32 | 126 | 74 |
| 199 | Male | 32 | 137 | 18 |
| 200 | Male | 30 | 137 | 83 |

What should the index of this dataframe be set to? : CustomerID

*You are correct! The answer “*CustomerID*” is correct because the index of a dataframe should be a unique value for every row.*

In the Python function “KMeans()”, the constructor ‘‘init’’ is used to select the criteria for the initialization of data points. : False

*You are correct! The answer “False” is correct because the constructor ‘init’ is used to select criteria for the initialization of the centroids of the clusters.*

In “KMeans()”, how do you generate the array that tells which cluster the data point belongs to? : kmeans.labels\_

*You are correct! The answer “*kmeans.labels\_*” is correct because the statement is used to get an array that tells which data point belongs to which cluster.*

The default initialization in K-means is random initialization. : False

*You are correct! The answer “False” is correct because the default initialization in K-means is improved initialization, which is K-means++.*

K-means++ only finds centroids once each time you run the function. : False

*You are correct! The answer “False” is correct because K-means++ finds the initial centroids and then searches again in an attempt to lower the inertia of the dataset.*

The number of clusters for DBSCAN are declared beforehand. : False

*You are correct! The answer “False” is correct because the DBSCAN algorithm is centroid-less and the number of clusters arise naturally from the algorithm.*

What is the clustering algorithm which has the ability to create curved boundaries between clusters? : DBSCAN

*You are correct! The answer “DBSCAN” is correct because it has the ability to create curved boundaries between clusters.*

Points that are sufficiently removed from other points are designated by DBSCAN as (blank). : Outliers

*You are correct! The answer “Outliers” is correct because DBSCAN has a built-in outlier detection feature*.*Points that are sufficiently removed from other points and not classified at all are called outliers.*

In the Python function “cluster.DBSCAN()”, the constructors of the function are (blank). *(Check all that apply.)* : min\_samples, eps

*You are correct! The answers “eps” and “min\_samples” are correct because these are the constructors for the function of DBSCAN.*

If the ball of radius epsilon captures less than min\_sample points, then that point is designated as a core point. : False

*You are correct! The answer “False” is correct because if the ball of radius epsilon captures at least min\_sample points, then that point is designated as a core point.*

How does DBSCAN declare a point as an outlier? : Points with no core or boundary points in their epsilon ball radius

*You are correct! The answer “Points with no core or boundary points in their epsilon ball radius” is correct because such data points that do not have any core or boundary point in their epsilon are declared as outliers.*

**Savio’s Session**

*sse = {}*

*for k in range(1,10):*

*kmeans = KMeans(n\_clusters = k, max\_iter = 1000).fit(df\_pca)*

*sse[k] = kmeans.inertia\_*

*plt.figure()*

*plt.plot(list(sse.keys()),list(sse.values()))*

*Feature vs categorical*

*df.groupby(‘cluster’).*

*Use standard scaler for try-it 6.2!*

*.agg(pd.Series.mode)*

**6.1: Summarizing Data with PCA - Section B**

For PCA analysis, we can take a look at the scree plot to decide how many components to keep. First, let’s normalize out dataset:

# normalize data

default\_norm = (default - default.mean()) / default.std()

We can set number of components to a large value first to analyze components visually to decide how many to keep:

# initialize to 10 components to visualize Scree Plot first

pca = PCA(n\_components = 10, random\_state = 42)

pca.fit\_transform(default\_norm)

Then, draw a scree plot to visualize components:

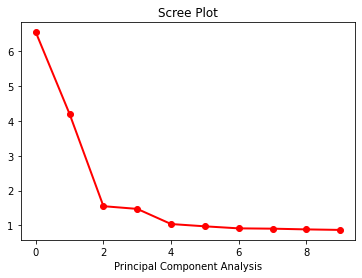
# Scree Plot

plt.plot(pca.explained\_variance\_, 'ro-', linewidth=2)

plt.title('Scree Plot')

plt.xlabel('Principal Component Analysis')

plt.show()



Most significant components are first 2, however, one may pick up to 4 components but I chose only first 2 for further scatter plot analysis:

# initialize to 2

pca = PCA(n\_components = 2, random\_state = 42)

components = pca.fit\_transform(default\_norm)

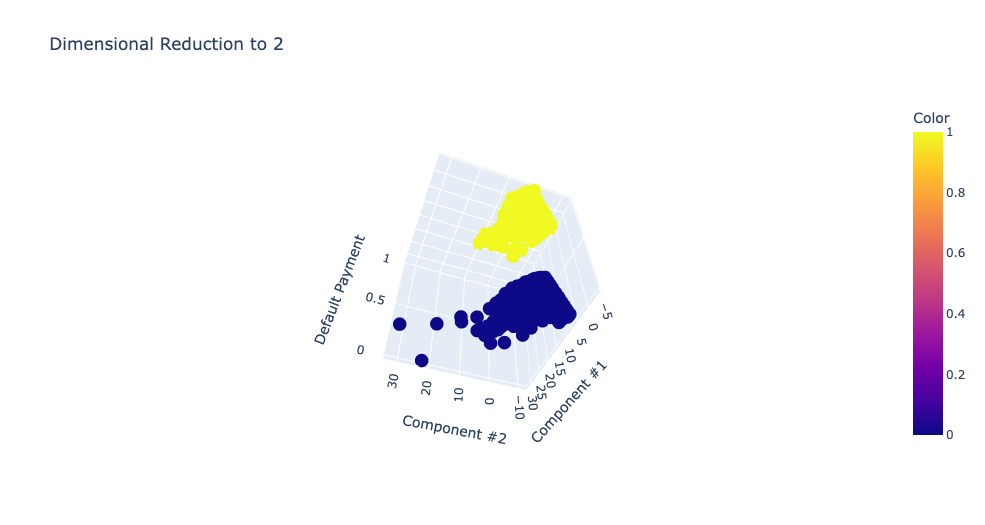
# Scatter plot by default.payment.next.month

px.scatter\_3d(x=components[:, 0], y=components[:, 1], z=default['default.payment.next.month'],

color=default['default.payment.next.month'],

title='Dimensional Reduction to 2',

labels={"z":"Default Payment", "x":"Component #1", "y":"Component #2", "color":"Color"})



As shown in the 3D scatter plot, it did not achieve the goal by r=2 since there is no a clear line between these two groups by default.payment.next.month, in fact, they are overlapping because ‘Default Payment’ is not significant enough by itself to classify data points close to each other per principle component. Also, these 2 principle components only represent 45% of dataset, but, I do not think representing more of the dataset would change the scatter plot.

# Cumulative sum of ratios

np.cumsum(pca.explained\_variance\_ratio\_)

array([0.272986 , 0.44804925, 0.51275778, 0.57423046, 0.61758316,

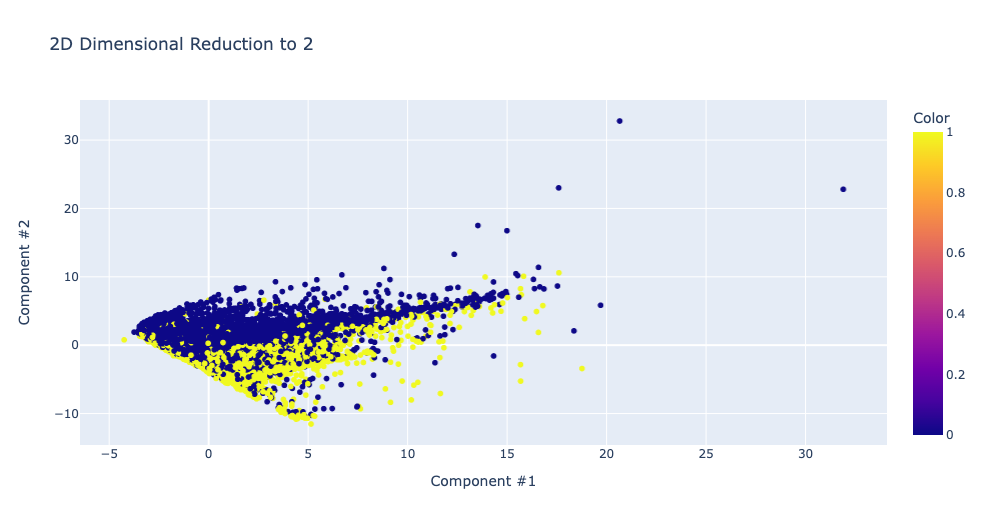
0.65818493, 0.69635315, 0.7341443 , 0.77106995, 0.8073719 ])

# Scatter plot by default.payment.next.month

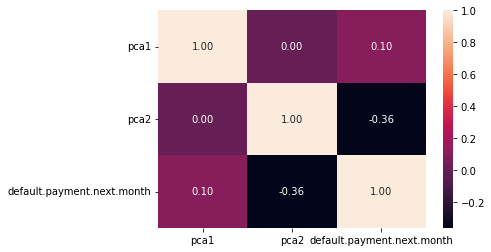
px.scatter(data\_frame=components, x=0, y=1, color=default['default.payment.next.month'],

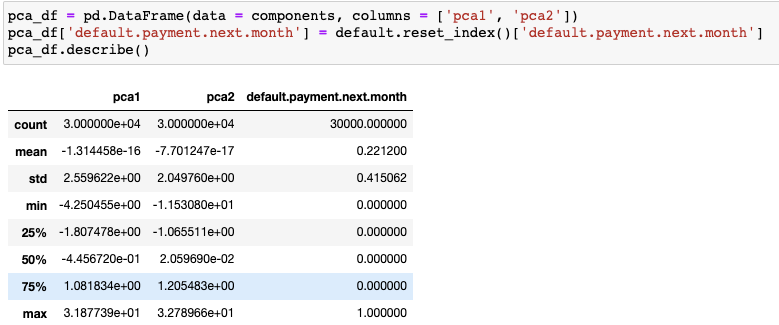
title='2D Dimensional Reduction to 2',

labels={"0":"Component #1", "1":"Component #2", "color":"Color"})



sns.heatmap(pca\_df.corr(), annot=True, fmt='.2f')





**6.2: Interpreting the Results of K-Means and PCA - Section B**

For PCA and clustering, first the dataset needs some cleaning and transformation as below.

**Data Preparation:**

* Transform all Yes/No columns to 1/0
* Transform Gender Male/Female to 1/0
* Analyze columns Offer, Internet Type, Contract, Payment Method, Churn Category and Churn Reason
* Fill null Customer Satisfaction with mean value 3.005453
* Fill null Churn Reason with ‘Don't know’
* Fill null Churn Category with ‘Other’
* Set index to Customer ID
* Drop column City since there is zip code and latitude & longitude
* Drop dependents since there is number of dependents
* Drop Under 30 and Senior Citizen since there is Age column

**Dimensional Reduction:**

Initialize r=20 to analyze PCA to decide how many to keep:

# initialize r to 20 components to visualize Scree Plot first

pca = PCA(n\_components = 20, random\_state = 42)

pca.fit\_transform(df\_norm)

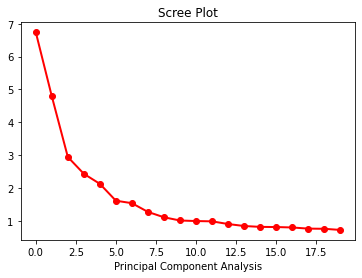
Scree plot

plt.plot(pca.explained\_variance\_, 'ro-', linewidth=2)

plt.title('Scree Plot')

plt.xlabel('Principal Component Analysis')

plt.show()



I checked cumulative ratios as well for 20:

# Draw a cumulative sum to decide how many to keep

np.cumsum(pca.explained\_variance\_ratio\_)

array([0.1644471 , 0.28131084, 0.35310188, 0.4124016 , 0.46424793,

0.50356417, 0.54105027, 0.5720512 , 0.59907129, 0.6236999 ,

0.64791467, 0.671911 , 0.69391559, 0.7144539 , 0.73434008,

0.75409989, 0.77354679, 0.79214295, 0.81060817, 0.82823602])

r=6 seems represents more than 50% of features in the dataset.

**Clustering Dataset:**

After running k-means up to 10 clusters by checking inertia, I chose 4 clusters to execute k-means:

# kmeans inertia check

inertias = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i, init='k-means++', random\_state=42).fit(components)

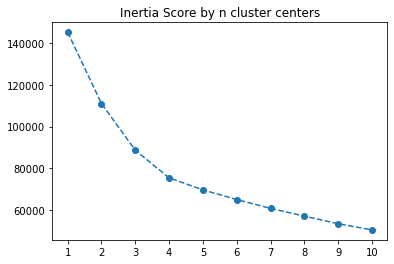
inertias.append(kmeans.inertia\_)

print(inertias)

plt.plot(list(range(1, 11)), inertias, '--o')

plt.xticks(list(range(1, 11)), list(range(1, 11)))

plt.title('Inertia Score by n cluster centers')



# kmeans

kmeans = KMeans(n\_clusters = 4, random\_state = 42).fit(components)

#df\_clean['cluster'] = kmeans.labels\_

np.unique(kmeans.labels\_)

**Results:**

There are many features in this dataset that one can find many different results, listing a few samples

**Total Charges:**

Cluster 3: Customer segment with total regular charges between $2000 and up, high long distance charges and no extra data charge.

Cluster 1: Customer segment with total regular charges up to $2000 and high long distance charges and no extra data charge.

Cluster 2: Customer segment with total regular charges up to $3000 and moderate long distance extra data charges.

Cluster 0: Customer segment with total regular charges up to $4000 and moderate long distance charges and no extra data charge.

px.scatter\_3d(data\_frame=df, x = 'Total Regular Charges',

y = 'Total Extra Data Charges', z = 'Total Long Distance Charges', color = 'kmeans',

title='Charges by Clusters',

labels={"kmeans":"Clusters"})



**Churn:**

Cluster 3: Customer segment with 2 year contract

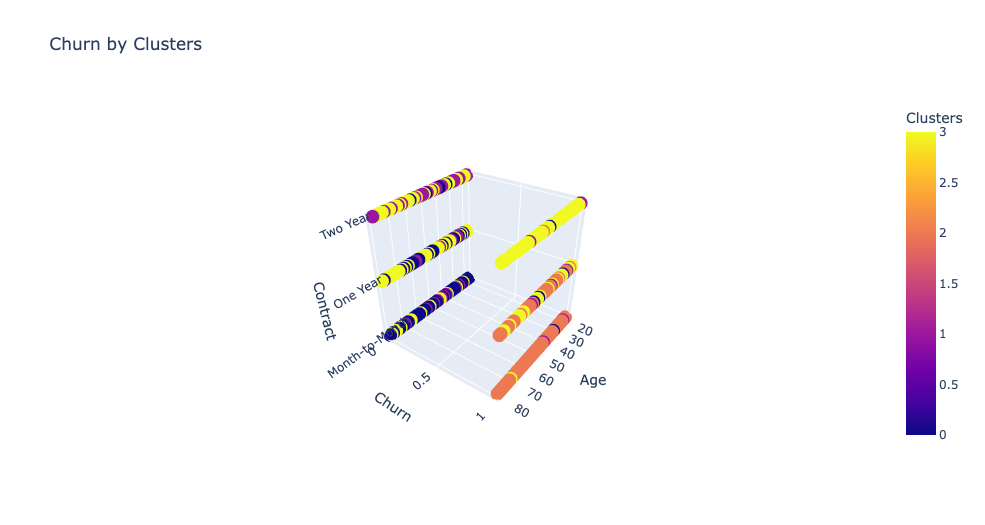
Cluster 2: Customer segment with month-to-month contract

Cluster 0: Customer segment who stays with month-to-month contract

px.scatter\_3d(data\_frame=df, x = 'Age', y = 'Churn Value', z = 'Contract', color = 'kmeans',

title='Churn by Clusters',

labels={"Churn Value":"Churn", "kmeans":"Clusters"})

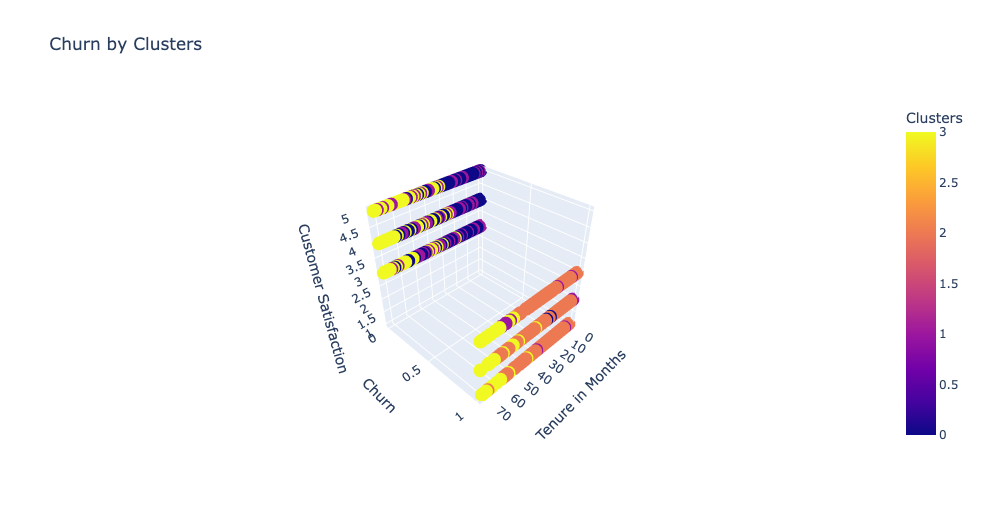


Also, Customer segment in Cluster 2 with month-to-month contract, points to low customer satisfaction which is 3 and below.

px.scatter\_3d(data\_frame=df, x = 'Tenure in Months', y = 'Churn Value', z = 'Customer Satisfaction', color = 'kmeans',

title='Churn by Clusters',

labels={"Churn Value":"Churn", "kmeans":"Clusters"})



**Total Spending:**

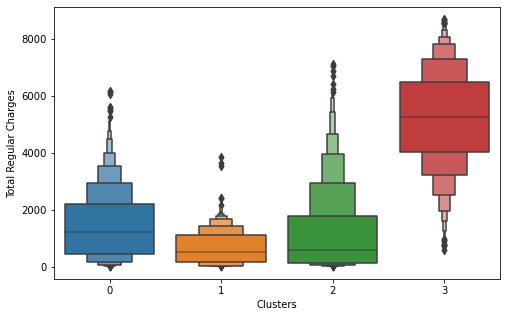
This is another angle about the customer segment in Cluster 3 which is clearly high-spending by buying all services and Cluster 1 is low-spending customer segment who buy phone service and nothing else.

plt.tight\_layout()

plt.subplots(figsize=(8,5))

sns.boxenplot(data=df, x='kmeans', y='Total Regular Charges')

plt.xlabel('Clusters')



**Break-down by Service:**

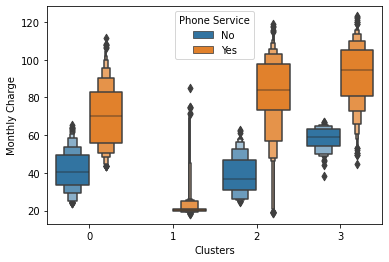
Customer segment in Cluster 1 goes with Phone Service only, they do not buy any other service, low paying customer segment.

plt.tight\_layout()

plt.subplots(figsize=(6,4))

sns.boxenplot(data=df, x='kmeans', y='Monthly Charge', hue='Phone Service')

plt.xlabel('Clusters')



**Overall Conclusion**

Checked overall dataframe by grouping it by cluster to analyze entire dataframe:

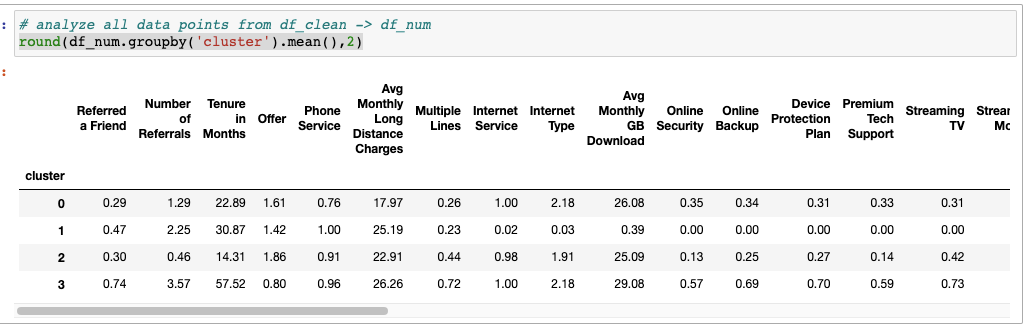
*Cluster 0* is total spending is second highest in this customer segment, lesser on phone service, on month-to-month, they go with unlimited data more, tend to be single, fewer dependents, goes with paperless billing.

*Cluster 1* is low-spending customer segment phone-only service, mid-age, tenure is longer, on typically 1 year contract, marital status is mixed, highest number of dependents, second most referrals, pay by credit card, goes with paper billing.

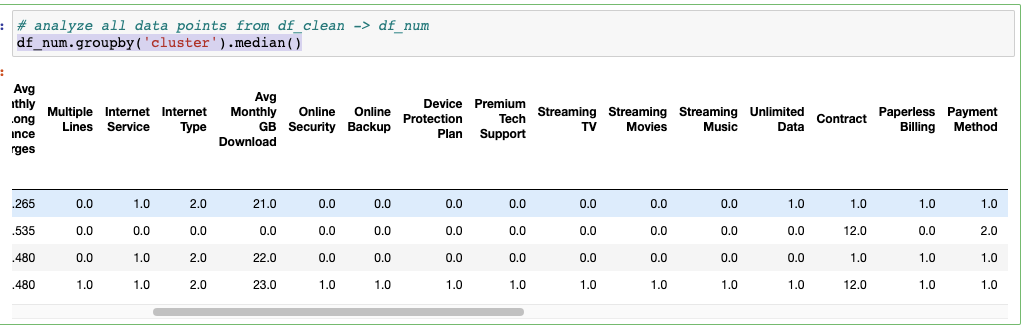
*Cluster 2* is high churn-rate due to attitude of support person, this segment ask most customer service requests by reporting product/service issues, low customer satisfaction, more mature folks, on month-to-month, tend to be single, none to fewer dependents, total spending is less due to short tenure but monthly charge is second highest, fewest referrals, pay by bank, goes with paperless billing.

*Cluster 3* is high-spending customer segment by buying all services, longest tenure in this segment, on typically 1 year contract, married, some dependents, they pay highest monthly charges, refer many friends, goes with paperless billing.

round(df\_num.groupby('cluster').mean(),2)



df\_num.groupby('cluster').median()



————— o —————

**Mentoring**

<https://ml-ops.org/>

<https://www.credly.com/skills/hadoop> -> is arama

<https://www.youtube.com/watch?v=Bw7nyOmvoe4>

**Resume Career Help:** [**danielle.fortier@emertius.org**](mailto:danielle.fortier@emertius.org)