G Varchaleswari,

vganugapati1@student.gsu.edu

2575320

Question 1 [5 Pts]

What is Unsupervised Learning and difficulties involved in unsupervied Learning and name a few unsupervised algorithms ?

Solution

Unsupervised learning is a branch of machine learning which deals with unknown data that comes with a set of features and has no associated target variable. That is to say, there exists no prior knowledge about the outcome of modelling. Unsupervised learning focuses on dividing the data into groups or clusters such that it lays an understanding of the similarity/dissimilarity between different data points rather than predicting any outcome. It is a measure of interesting associations between different features such as determining subgroups and their intra cluster behaviour; determinng the differences between two groups or subgroups by measuring the inter cluster distances (any Euclidean or relevant distance measures).

Unsupervised learning comes with a drawback - not having a target variable. Now, if we understand that unsupervised learning is all about subjective analysis of data, it needs extensive study of data by different visulaization techniques and the statistical tools available at our disposal. This task becomes very difficult when there is no gaurantee or check measure available to verify the analysis. Imagine describing an unknown data and not having an idea of how good is your description or analysis. That is because there is no standardization of desired outcomes rather it is based on assumptions that are drawn out of visualization and statistical techniques applied to the data. That means, we will always need to do great hard work in describing the data because the analysis is based on few assumptions which may or may not be true.

Few unsupervised learning algorithms are - Principal Component Analysis, K-means clustering, KNN clustering, Hirarchical clustering.

Question 2 [5 Pts]

What is a PCA [Principal Component Analysis] and When to use PCA [Principal Component Analysis] ?

Solution

PCA is an unsupervised approach that is used to compute the principal components for a given dataset and the components are subsequently used for understanding the data. Principal components are the directions or vectors in a feature space along which the original data has most of the variance.

Imagine we are presented with a dataset with some hundreds of feature vectors, could that slow down or infact confuse our machine learning algorithm ? Yes, indeed, it will be a difficult task to handle so many features while traning the model. What if the features in the dataset have a relationship with the observations such that one feature is more dominant in decribing the data as compared to other features in the dataset? That means we can use the feature which decribes the data almost clearly (almost because it usually does not decribe 100% of the outcome) and avoid the ones which have least significance in the decision. This will help to reduce the overhead on the model. This task of summarizing the dataset with a smaller number of representative variables that collectively explain most of the variability in the original dataset is termed as principal component analysis.

Question 3 [5 Pts]

How does a PCA work [Principal Component Analysis] ?

Solution

PCA finds a low dimensional space for the given dataset with as much variance as possible. The idea is that each of the n observations lives in p-dimensional space, but not all of these dimensions are equally interesting. The measure of intrestingness is

observations lives in p-dimensional space, but not all of these dimensions are equally interesting. The measure of intresting is the amount of variation each observation has in the different dimensions. If one particular dimension preserves almost 70% of the varaiance in data which is the highest for that data on the given dimension, the same dimension can be used to represent the observations with 70% of the varaince. This dimension would be the principal component or the eigen vector corresponding to the largest eigen value of the data. It is also the first principal component of the data.

PCA works by calculating the average measurements of the data with respect to features and finds a centre point. Now the data is shifted such that the centre lies on top of the origin(0,0) in the coordinate plane. Lets call this center 'C'. Relative positions of the data points are always preserved. Now, we fit a line that passes through the point 'C'. The line is rotated until the appropriate position of the line that fits best to the data is found. Once the line is found, the projected data points are the representation of the observations on the given component.

PCA works by taking projections of data points on a given component and then it either measures the distances from the data to the line and tries to find the line that minimizes those distances or it can try to find the line that maximizes the distances from the projected points to the origin.

Now lets look at the maths behind the same. The first principal component of a set of features X1, X2,...,Xp is the normalized linear combination of the features that has the largest variance.

(Ref: An Introduction to Statistical Learning, Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani)

Z1 = φ11 X1 + φ21 X2 + ... + φp1 Xp

By normalized, we mean that

Summation over square(φj1) = 1

We refer to the elements φ11,...,φp1 as the loadings of the first principal component. The loadings make up the principal component loading vector, φ1 = (φ11 φ21 ... φp1)T .

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

The aim is to maximize the objective function that is defined above. This set of loading vectors and the maximization of the equation presented above can be found by eigen value decomposition. That helps in finding all the eigen values for a given data along with their corresponding eigen vectors such that the largest eigen vector corresponds to the largest eigen value. This largest eigen vector is the first principal component that retains the maximum possible variance on a single dimension of the dataset.

Question 4 [5 Pts]

What are different methods by which you can compute the PCA? Does every method will yield same result or answer at the end?

Solution

There exist two different ways that can be used to compute PCA:

1) Eigen value Decomposition

The PCs can be determined via eigen decomposition of the covariance matrix C (X*X.transpose) of the data matrix X. Eigen decomposition, in fact, finds a new coordinate system of the eigenvectors for C through various rotations.

C = E D E.transpose

Matrix of eigenvectors E(m x m) and a diagonal matrix of "m" eigenvalues D is the eigen decomposition of C (m x m). The eigenvectors or column vectors in E are the PCs we wish to evaluate. We then use matrix multiplication the data projection onto the principal components.

2) Singular Value Decomposition

Consider the same representation of covariance matrix stated above. "C*C.transpose" is U and "C.transpose*C" is V such that U and V are orthogonal matrices, i.e. "V*V.transpose = I(indentity matrix)" and "U*U.transpose = I(indentity matrix)".

By SVD rule:

C = U S V.Transpose

where S matrix is a compositon of singular values found by application of square root onto the eigen values of U and V such that all the singular values are arranged diagonally in descending order.

Yes, the aim of both the methods is the same, i.e. to find the data in the directions with the highest variances. Yet, SVD is considered more stable and suited for complex, square or rectangular matrices of data while eigen decomposition can only work with square matrices.

Question 5 [5 Pts]

What are advantages and disadvantages of PCA[Explain with example]?

Solution

Advantage of PCA is that it reduces the higher dimensions of data to lower dimensions such that the most of the variance is preserved and the overhead of compuation for a machine learning algorithm decreases with the reduction in dimensionality of data. Usually, the information in a data set is concentrated in its first few principal components and noise, if any, tend towards the later principal components. The same is applicable when we wish to visualize the relation between different features, it becomes highly impossible to visualize and infer from data that has high dimensions. PCA helps in visualizing the higher dimensional data in lower dimensions. Consider a motor insurance fraud detection system which tries to predict how claims made by different customers could be fradulent. In this the dataset may contain, name of the customer, details about previous claims, amount claimed, marital status, injury type of the vehicle, and gender, etc., amongst which few features say like marital status or gender may not be of great importance in the prediction. PCA could help reduce the dimensions of data such that relevant predictions are made.

Disadvantage of PCA is that it highly relies on the approximation of points, i.e. the projection of data points in lower dimensions that result in some loss of information in data. That means the original independent features are no more existing rather the principal components have created a new derived features from the different correlated features. Standardization is a compulsory step before PCA as the features with high values can influence the variance of data on the respective components.

Question 6 [10 Pts]

What is clustering? Explain how K-Means Clustering Algorithm works?

Solution

Clustering is an unsupervised way of grouping data into different groups. Observations in same subgroups share some similarity amongst the observations within them and are different from the observations from different subgroups.

K-means clustering works by dividing the dataset into non-overlapping subgroups such that the intra cluster distances are minimized and the inter cluster distances are maximized.

1 - The number of clusters are choosen.

2 - Each cluster must contain one centroid. Centroid is the mean value of the observations in a cluster i.e the centre point of the set of observations in a cluster.

3 - Initially, since the clusters are not known, centroids are assigned at random.

4 - Then each observation's distance from all the centroids is evaluated. The observation is assigned to a cluster for which the centroid has the least value of the distance to the given observation

5 - After all the observations are assigned to clusters, mean value of the cluster is evaluated. The new centroid is updated to the mean value

6 - Then the steps 4 and 5 are followed again until the algorithm converges to a local minimum. i.e. when the centroids are no more changing. This is the point when the clusters are defined with clear boundaries.

Question 7 [10 Pts]

What are the Advantages and disadvantages of Clustering ALgorithms discussed in our class (K-Means,Hierchal)?

Solution

Advantages of hierarchical and k-means is that it is relatively simple to understand and implement and guarantees convergence. The dendrogram produced after applying hierarchical clustering can be used to understand the bigger picture along with the subgroups of the data. Clustering is very useful in detecting the subgroups among the data.

In k-means clustering deciding upon K even before starting the clustering is the main disadvantage for k-means. In k-means or hirarchichal clustering, the given observation or data point can belong to only one cluster i.e. even if the data might share similarity with other clusters, the clustering algorithm constraints the set of points to be at only one cluster and not within multiple clusters.

Question 8 [5 Pts]

Which Clustering Algorithm is better K-Means or hierarchical Clustering? Explain with a proper example which is better algorithm?

Solution

Hirarchical clustering is any time better than K-means, the biggest factor being choice of cluster number must be given even before the k-means algorithm starts training. Unlike k-means, hirarchical clustering doesnt need a mention of clusters before the process starts. For example consider a dataset of covid-19 outbreak that consists the details of outbreak across the globe. We intend to study the most affected cluster groups and their impact on the spread of desease across the globe. K-means demands to give the required number of clusters before, that in a way is a problem because our intention is to cluster the data based on the simmilarity or dissimilarity measure within it without any prior knowledge of how many clusters should or could be present. In such case, hirarchical clustering helps in not only finding the subgroups without the mention of clusters initially, but also defines a tree like structure i.e. a dendogram.

Question 9 & 10 [50 Pts]

Please Perform Principal Component Analysis and K-Means Clustering on the Give dataset Below.

10 Points for Data Preprocessing.

15 Points for PCA Algorithm along with plots and Results Explaination.

15 Points for K-Means Algorithm with plots and Results Explination.

10 Points for Comparing the results between PCA and K-Means and whats your inference from your ouputs of the algorithms.

Hints:

1. As per the data preprocessing step convert all the variables in the dataset into Numerical values as the algorithms only work with Numerical values
2. Then Apply both algorithms one after the other then plot the ouput clusters
3. Compare the output clusters in both the steps.

Bank Marketing Data Set

URL: https://dataminingcsc6740.s3-us-west-2.amazonaws.com/datasets/homework_2.csv

Files Information:

the homework_2.csv has all the informatiom.

Relevant Information:

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed.

There are two datasets: 1) The csv file has 4521 data points in it. 2) Number of Attributes: 16 + output attribute.

Attribute information:

Input variables:

bank client data:

1 - age (numeric)

2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar","self-employed","retired","technician","services")

3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)

4 - education (categorical: "unknown","secondary","primary","tertiary")

5 - default: has credit in default? (binary: "yes","no")

6 - balance: average yearly balance, in euros (numeric)

7 - housing: has housing loan? (binary: "yes","no")

8 - loan: has personal loan? (binary: "yes","no")

related with the last contact of the current campaign:

9 - contact: contact communication type (categorical: "unknown","telephone","cellular")

10 - day: last contact day of the month (numeric)

11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")

12 - duration: last contact duration, in seconds (numeric)

other attributes:

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)

15 - previous: number of contacts performed before this campaign and for this client (numeric)

16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")

Output variable (desired target):

17 - y - has the client subscribed a term deposit? (binary: "yes","no")

In [44]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

In [71]:

```
data = pd.read_csv('homework_2.csv',delimiter=';')
data.head()
```

Out[71]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0

Made a copy of the pandas dataframe such that the categorical varaibles align one after other. This helps in preprocessing the data easily and I do not want to manipulate the original data that was imported along with the target variable. Since unsupervised learning deals with only features of the data without any prior information on the target variable, I seperate the target variable from the dataset.

In [72]:

```
x = pd.DataFrame(index=range(data.shape[0]), columns=['age','balance','day','duration','campaign','
pdays','previous','job', 'marital', 'education', 'default', 'housing',
       'loan', 'contact', 'poutcome','month'])
x = pd.DataFrame(data, dtype=None, copy=False, index = x.index, columns=x.columns )
y = pd.DataFrame(data, dtype=None, copy=False, index = x.index, columns=['y'] )
x.head()
```

Out[72]:

	age	balance	day	duration	campaign	pdays	previous	job	marital	education	default	housing	loan	contact	poutcc
0	30	1787	19	79	1	-1	0	unemployed	married	primary	no	no	no	cellular	unkn
1	33	4789	11	220	1	339	4	services	married	secondary	no	yes	yes	cellular	fai
2	35	1350	16	185	1	330	1	management	single	tertiary	no	yes	no	cellular	fai
3	30	1476	3	199	4	-1	0	management	married	tertiary	no	yes	yes	unknown	unkn
4	59	0	5	226	1	-1	0	blue-collar	married	secondary	no	yes	no	unknown	unkn

In [47]:

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 16 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        4521 non-null   int64 
 1   balance    4521 non-null   int64 
 2   day        4521 non-null   int64 
 3   duration   4521 non-null   int64 
 4   campaign   4521 non-null   int64 
 5   pdays      4521 non-null   int64 
 6   previous   4521 non-null   int64 
 7   job        4521 non-null   object
 8   marital    4521 non-null   object
 9   education  4521 non-null   object
 10  default    4521 non-null   object
 11  housing    4521 non-null   object
 12  loan       4521 non-null   object
 13  contact    4521 non-null   object
 14  poutcome   4521 non-null   object
 15  month      4521 non-null   object
dtypes: int64(7), object(9)
memory usage: 565.2+ KB
```

No null values in the dataset

In [48]:

```
x.isnull().sum()
```

Out[48]:

```
age          0
balance      0
day          0
duration     0
campaign     0
pdays        0
previous     0
job          0
marital      0
education    0
```

```
education    0
default      0
housing      0
loan         0
contact      0
poutcome     0
month        0
dtype: int64
```

There are no outliers in the dataset

In [73]:

```
print(x.loc[x['age']<18])
print(x.loc[x['day']<0])
print(x.loc[x['duration']<0])
print(x.loc[x['campaign']<0])
print(x.loc[x['pdays']<-1])
print(x.loc[x['previous']<0])
```

```
Empty DataFrame
Columns: [age, balance, day, duration, campaign, pdays, previous, job, marital, education,
default, housing, loan, contact, poutcome, month]
Index: []
Empty DataFrame
Columns: [age, balance, day, duration, campaign, pdays, previous, job, marital, education,
default, housing, loan, contact, poutcome, month]
Index: []
Empty DataFrame
Columns: [age, balance, day, duration, campaign, pdays, previous, job, marital, education,
default, housing, loan, contact, poutcome, month]
Index: []
Empty DataFrame
Columns: [age, balance, day, duration, campaign, pdays, previous, job, marital, education,
default, housing, loan, contact, poutcome, month]
Index: []
Empty DataFrame
Columns: [age, balance, day, duration, campaign, pdays, previous, job, marital, education,
default, housing, loan, contact, poutcome, month]
Index: []
Empty DataFrame
Columns: [age, balance, day, duration, campaign, pdays, previous, job, marital, education,
default, housing, loan, contact, poutcome, month]
Index: []
```

Descriptive Statistics for the dataset

In [74]:

```
x.describe()
```

Out[74]:

	age	balance	day	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.542579
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.693562
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.000000
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.000000
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.000000
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.000000
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.000000

Encoding

Encoding

The categorical columns in the dataset should be changed to numeric values for further processing of data using PCA or K-Means

In [75]:

```
x['job'] = x['job'].astype('category')
x['marital'] = x['marital'].astype('category')
x['education'] = x['education'].astype('category')
x['default'] = x['default'].astype('category')
x['housing'] = x['housing'].astype('category')
x['loan'] = x['loan'].astype('category')
x['contact'] = x['contact'].astype('category')
x['poutcome'] = x['poutcome'].astype('category')
x['month'] = x['month'].astype('category')

x['job'] = x['job'].cat.codes
x['marital'] = x['marital'].cat.codes
x['education'] = x['education'].cat.codes
x['default'] = x['default'].cat.codes
x['housing'] = x['housing'].cat.codes
x['loan'] = x['loan'].cat.codes
x['contact'] = x['contact'].cat.codes
x['poutcome'] = x['poutcome'].cat.codes
x['month'] = x['month'].cat.codes
x.dtypes
```

Out[75]:

```
age           int64
balance       int64
day           int64
duration      int64
campaign      int64
pdays         int64
previous      int64
job            int8
marital        int8
education      int8
default        int8
housing        int8
loan           int8
contact        int8
poutcome       int8
month          int8
dtype: object
```

In [52]:

```
x.head()
```

Out[52]:

	age	balance	day	duration	campaign	pdays	previous	job	marital	education	default	housing	loan	contact	poutcome	mont
0	30	1787	19	79	1	-1	0	10	1	0	0	0	0	0	3	
1	33	4789	11	220	1	339	4	7	1	1	0	1	1	0	0	
2	35	1350	16	185	1	330	1	4	2	2	0	1	0	0	0	
3	30	1476	3	199	4	-1	0	4	1	2	0	1	1	2	3	
4	59	0	5	226	1	-1	0	1	1	1	0	1	0	2	3	

Scaling is important before doing PCA

In [76]:

```
from sklearn.preprocessing import StandardScaler
#Feature Scaling using StandardScaler
x_scaled = StandardScaler().fit_transform(x)
```

In [77]:

```
x_scaled = pd.DataFrame(x_scaled, dtype=None, copy=False, index = x.index, columns=x.columns)
```

In [78]:

```
x_scaled.head()
```

Out[78]:

	age	balance	day	duration	campaign	pdays	previous	job	marital	education	default	housing	loan
0	-1.056270	0.121072	0.374052	-0.711861	-0.576829	-0.407218	-0.320413	1.716804	-0.246429	-1.644755	-0.130759	-1.142051	-0.424756
1	-0.772583	1.118644	-0.596026	-0.169194	-0.576829	2.989044	2.041734	0.795246	-0.246429	-0.309038	-0.130759	0.875617	2.354292
2	-0.583458	-0.024144	0.010273	-0.303898	-0.576829	2.899143	0.270124	-0.126313	1.421396	1.026680	-0.130759	0.875617	-0.424756
3	-1.056270	0.017726	-1.566105	-0.250017	0.387967	-0.407218	-0.320413	-0.126313	-0.246429	1.026680	-0.130759	0.875617	2.354292
4	1.686036	-0.472753	-1.323585	-0.146102	-0.576829	-0.407218	-0.320413	-1.047871	-0.246429	-0.309038	-0.130759	0.875617	-0.424756

In [13]:

```
from sklearn.decomposition import PCA

pca = PCA()
principal = pca.fit_transform(x_scaled)

principal_component = pd.DataFrame(pca.components_.T , index = x_scaled.columns
            , columns = ['PC1',
'PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','PC12','PC13','PC14','PC15','PC16'])
principalDf = pd.DataFrame(principal, index = x_scaled.index, columns = principal_component.columns
)
principalDf
```

Out[13]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC1
0	-0.559055	0.200792	-0.257270	0.471856	-0.048022	-0.543709	-0.070120	-0.434223	-0.251383	-1.336617	1.953777	0.629883	2.09021
1	4.499353	-1.097886	-0.197939	-0.448415	-0.355182	-0.897951	-0.455845	2.278541	-0.642141	0.259963	0.860522	0.890620	0.27978
2	3.725206	1.005552	-1.258858	-0.288959	0.084494	0.303204	-0.497391	-0.491094	0.093413	0.074479	-0.717494	0.703734	1.64228
3	-0.896786	-1.233641	-1.006536	-0.331992	-0.922975	-0.813984	-0.907970	1.670322	-1.320527	1.517273	-0.605514	-0.246470	0.71429
4	-0.861070	-2.155031	1.077449	0.887522	-0.440975	0.077885	-0.513613	-0.778932	-0.284114	0.351036	-0.370520	-0.649996	0.60111
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4516	-0.642672	0.481831	-0.732203	-1.064085	0.925759	0.201408	1.331078	-0.455945	-0.157580	-0.312528	0.130161	1.262114	0.47848
4517	-1.241401	-1.796171	0.337050	-2.423779	-4.353185	-5.371289	0.933258	0.231590	3.367560	0.820363	-0.237642	-0.683979	1.18366
4518	-0.902169	2.515928	1.460930	-0.993725	1.249220	-0.399207	-0.454651	-0.792221	-0.422407	1.410286	1.102551	-0.316034	0.11198
4519	2.791317	0.556969	0.104107	-0.560394	-0.081888	0.574746	-1.378392	-0.727984	0.191547	0.544664	-0.230529	-0.834288	0.23600
4520	4.431982	0.409177	-0.393277	-1.127994	-1.268991	0.357678	-1.124084	1.690566	-0.636902	1.218138	-0.378138	-0.666761	1.96543

4521 rows × 16 columns

In [14]:

```
principal_component
```

Out[14]:

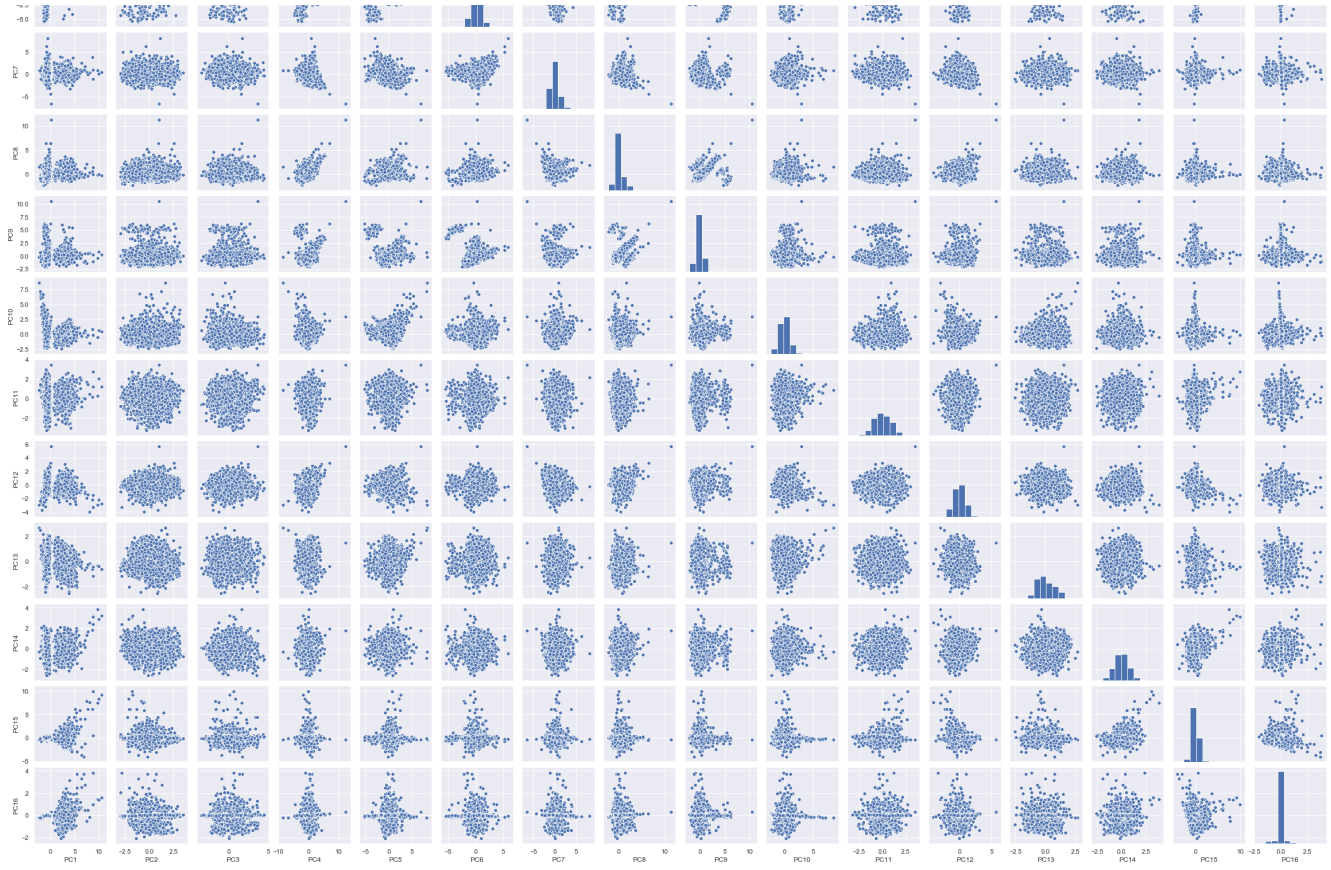
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	
age	-0.022632	0.011786	0.682792	0.137365	0.056168	-0.099465	0.094122	0.002644	-0.007049	0.002458	0.047472	-0.148874	0.1
balance	0.028044	0.062636	0.075438	0.484429	0.328278	0.003805	-0.250394	0.510867	0.453637	0.147430	0.095914	0.269633	0.0
day	-0.087497	0.121145	-0.045508	-0.298530	0.500286	-0.031832	0.451296	0.137962	0.158922	-0.591704	-0.010317	0.126735	0.1
duration	0.017975	-0.031237	-0.005279	0.114040	-0.314591	0.556806	0.658932	0.236656	0.169269	0.222578	0.020668	-0.080103	0.0
campaign	-0.109448	0.130601	-0.015231	-0.397037	0.515012	-0.032123	0.086848	-0.001667	0.017838	0.653634	0.108609	-0.258331	0.1
pdays	0.567329	-0.073961	0.030516	-0.057156	0.047057	-0.032929	0.026720	-0.024959	-0.002859	0.035266	0.021366	-0.011279	0.0
previous	0.495564	-0.043253	0.021060	-0.021715	0.066541	-0.055783	0.063794	0.030212	0.010491	0.000953	0.120485	-0.194347	0.0
job	0.003647	0.310755	-0.120385	0.235186	-0.099596	-0.422231	0.352923	-0.026170	-0.315661	0.118052	0.549966	0.318405	0.0
marital	0.048835	0.165901	-0.593712	0.027628	-0.012364	0.184796	-0.212855	0.042056	0.152663	-0.107401	0.322564	-0.279385	0.1
education	0.046020	0.282850	-0.270185	0.272201	0.015640	-0.354415	0.198263	0.175469	-0.126448	0.119974	-0.685220	-0.218319	0.1
default	-0.035973	-0.037411	-0.013150	-0.269979	-0.383157	-0.487648	0.097185	-0.098802	0.715090	0.086948	0.017213	0.000065	0.0
housing	0.057590	-0.480990	-0.245322	-0.116771	0.108142	0.011555	0.081906	-0.049859	-0.062636	0.256654	-0.190592	0.614395	0.1
loan	-0.033647	-0.072543	0.050541	-0.431790	-0.273036	-0.135611	-0.146666	0.772695	-0.283662	-0.013046	0.070141	-0.016988	0.0
contact	-0.249225	-0.467078	-0.058618	0.187410	0.102923	-0.154903	0.085839	-0.028118	-0.070400	0.076364	0.196975	-0.314969	0.6
poutcome	-0.582151	0.051765	-0.037778	0.040199	-0.060815	0.044778	-0.026668	-0.002634	0.003357	0.005927	-0.024601	0.029649	0.0
month	-0.005081	-0.541237	-0.128993	0.207566	0.101412	-0.231204	0.152787	0.138145	-0.013192	-0.175467	0.065281	-0.270311	0.6

Visualization of Data on Principal Components

Lets visualize the datapoints along different combination of PCs such that a scatter plot can describe the 2-dimensinal spread of the data across the principal components.

In [15]:

```
import seaborn as sns
sns.set()
sns.pairplot(principalDf,diag_kind='hist', height = 2.0)
plt.show()
```



By looking at the scatter plot above, we see that the data points are too closely represented in every combination. This shows how just 1 or 2 PCs won't be enough to represent the entire information in the data. There will be a great loss of variance if only 2 PC's are to be considered.

PCs Variance

Now that we have created our required principal componenets, we must look at the variance values of the PCs so that we get to decide how many PCs actually account for the best cumulative preservation of the information (variance) provided by the data.

In [16]:

```
principal_variance = pd.DataFrame(pca.explained_variance_ratio_*100 , columns = ['% Variance
Preserved']
             , index = ['PC1', 'PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','PC12',
'PC13','PC14','PC15','PC16'])
principal_variance['Cumulative % Variance'] = principal_variance['% Variance Preserved'].cumsum()
principal_variance
```

Out[16]:

	% Variance Preserved	Cumulative % Variance
PC1	15.900594	15.900594
PC2	10.569819	26.470413
PC3	9.186163	35.656576
PC4	7.430061	43.086638
PC5	7.133231	50.219869
PC6	6.466161	56.686030
PC7	6.074001	62.760031
PC8	5.817690	68.577721
PC9	5.723607	74.301327
PC10	5.314670	79.615998
PC11	5.179714	84.795711
PC12	4.740918	89.536630

	% Variance Preserved	Cumulative % Variance
PC13	3.411729	92.948359
PC14	3.346899	96.295257
PC15	2.860505	99.155762
PC16	0.844238	100.000000

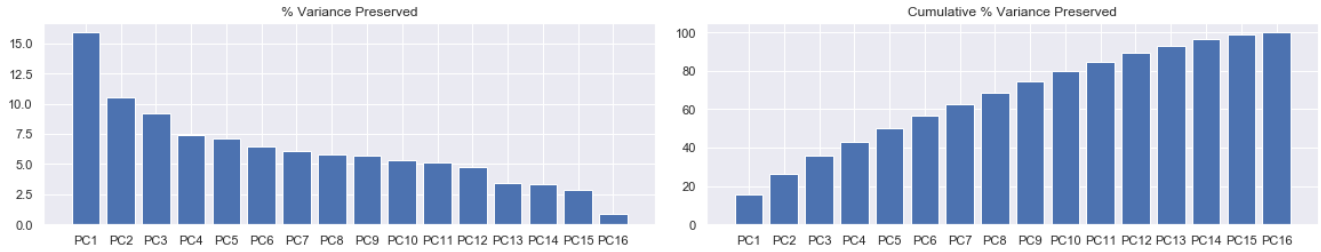
In [17]:

```
fig, ax = plt.subplots(1,2,squeeze=False,constrained_layout=True,figsize=(16,3))

ax[0][0].bar(principal_variance.index,principal_variance['% Variance Preserved'])
ax[0][0].set_title('% Variance Preserved')

ax[0][1].bar(principal_variance.index,principal_variance['Cumulative % Variance'])
ax[0][1].set_title('Cumulative % Variance Preserved')

plt.show()
```



Find best PCs

Now, looking at the bar graphs mentioned above and the elbow graphs presented below, we can see that PC1-PC11(inclusive) the variance percentage is over 80 and thus, the 16 dimensional data can be represented using these 11 principal components such that the maximum variance is preserved without losing much on the data.

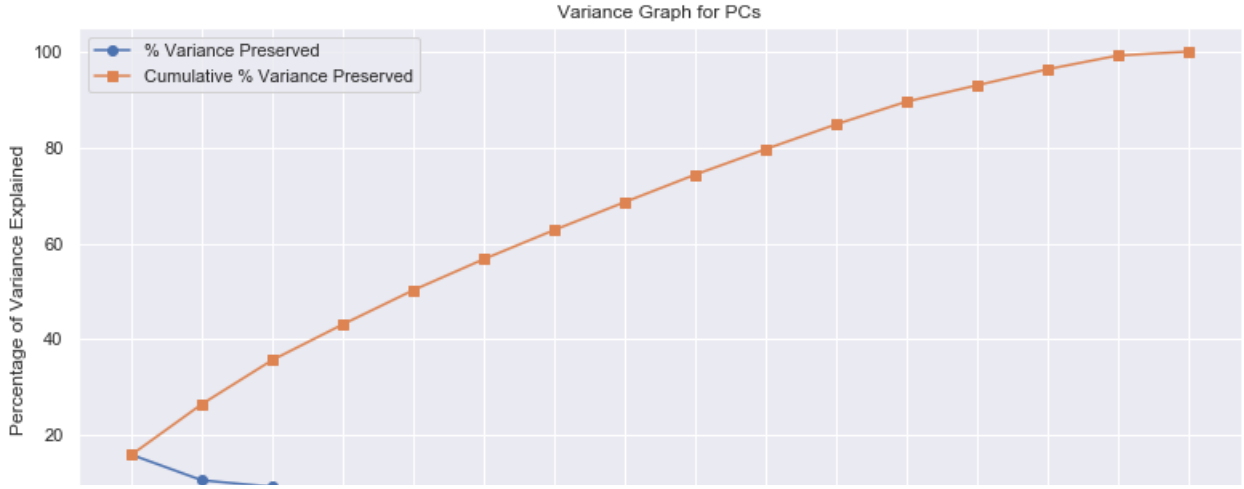
However, the choice of cumulative variance depends on the business requirement. The choice should be made such that the cumulative variance crosses the required threshold (For eg 80% and above).

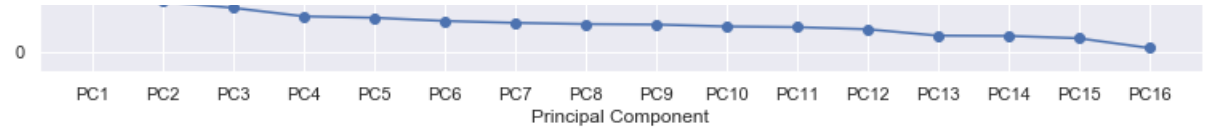
Here for understanding purpose, I mentioned PC1-PC11 will be a good choice.

In [18]:

```
plt.figure(figsize=(13,6))
plt.plot(principal_variance.index, principal_variance['% Variance Preserved'], '-o')
plt.plot(principal_variance.index, principal_variance['Cumulative % Variance'], '-s')
plt.xticks(principal_variance.index)
plt.ylabel('Percentage of Variance Explained')
plt.xlabel('Principal Component')
plt.legend(['% Variance Preserved','Cumulative % Variance Preserved'])
plt.title('Variance Graph for PCs')

plt.show()
```





Let us look at an example which obtains the first 11 PCs that preserve over 80% variance in data

In [19]:

```
pca_ex = PCA(n_components = 11)

principal_ex = pca_ex.fit_transform(x_scaled)

principal_component_ex = pd.DataFrame(pca_ex.components_.T , index = x_scaled.columns , columns = [
'PC1', 'PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11'])
principalDf_ex = pd.DataFrame(principal_ex, index = x_scaled.index, columns =
principal_component_ex.columns)
principalDf_ex
```

Out[19]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
0	-0.559055	0.200792	-0.257270	0.471856	-0.048022	-0.543709	-0.070120	-0.434223	-0.251383	-1.336617	1.953777
1	4.499353	-1.097886	-0.197939	-0.448415	-0.355182	-0.897951	-0.455845	2.278541	-0.642141	0.259963	0.860522
2	3.725206	1.005552	-1.258858	-0.288959	0.084494	0.303204	-0.497391	-0.491094	0.093413	0.074479	-0.717494
3	-0.896786	-1.233641	-1.006536	-0.331992	-0.922975	-0.813984	-0.907970	1.670322	-1.320527	1.517273	-0.605514
4	-0.861070	-2.155031	1.077449	0.887522	-0.440975	0.077885	-0.513613	-0.778932	-0.284114	0.351036	-0.370520
...	...	...	...	...	...	...	...	...	...	...	...
4516	-0.642672	0.481831	-0.732203	-1.064085	0.925759	0.201408	1.331078	-0.455945	-0.157580	-0.312528	0.130161
4517	-1.241401	-1.796171	0.337050	-2.423779	-4.353185	-5.371289	0.933258	0.231590	3.367560	0.820363	-0.237642
4518	-0.902169	2.515928	1.460930	-0.993725	1.249220	-0.399207	0.454651	-0.792221	-0.422407	1.410286	1.102551
4519	2.791317	0.556969	0.104107	-0.560394	-0.081888	0.574746	-1.378392	-0.727984	0.191547	0.544664	-0.230529
4520	4.431982	0.409177	-0.393277	-1.127994	-1.268991	0.357678	-1.124084	1.690566	-0.636902	1.218138	-0.378138

4521 rows × 11 columns

This summarizes the Principal component analysis for dimensionality reduction

K-Means

Visualizing correlation between features

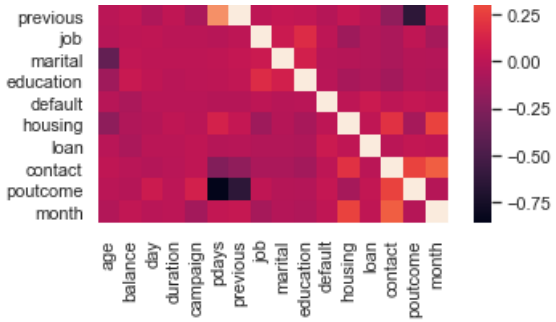
The correlation between different variables(features) can be best explained using heatmaps. The same is plotted below. A lighter shade represents high correlation and darker shade represents less correlation.

In [79]:

```
cor = x_scaled.corr()
sns.heatmap(cor, square = True) #Plot the correlation as heat map
```

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ceef8bc448>
```

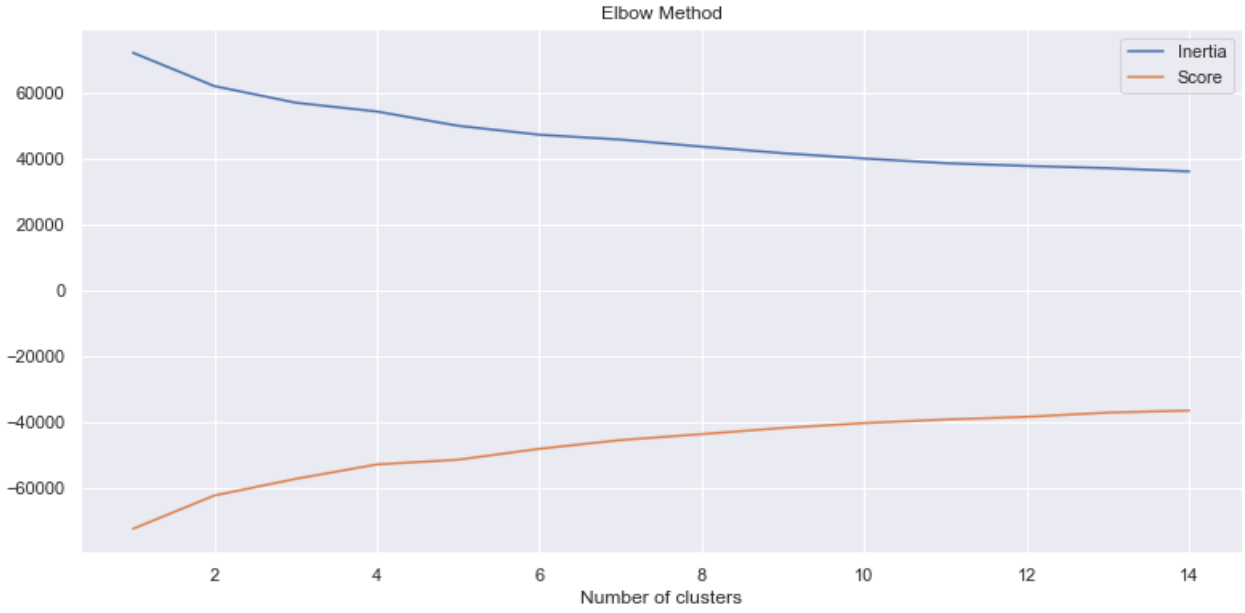


In [80]:

```
from sklearn.cluster import KMeans
def computeKMeans(samples, n=2,max_iter=300):
    km = KMeans(n_clusters = n,max_iter=max_iter)
    km.fit(samples)
    clusters = km.predict(samples)
    centroids = km.cluster_centers_
    inertia = km.inertia_
    score = km.fit(samples).score(samples)
    return (clusters, score, inertia)


iner = []
sc = []
for i in range(1,15):
    clusters, score, inertia = computeKMeans(x_scaled, n=i, max_iter=400)
    iner.append(inertia)
    sc.append(score)

plt.figure(figsize=(13,6))
plt.plot(range(1,15),iner)
plt.plot(range(1,15),sc)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.legend(['Inertia','Score'])
plt.show()
```



Choice of number of clusters based on the elbow graph

Based on the above-mentioned elbow graph analysis that explains the inertia with respect to the number of clusters, it is evident that the k-means algorithm's rate of decrease in inertia (Sum of squared distances of samples to their closest cluster center) decreases as the number of clusters increase beyond 6. For example, cluster number 2 has approximately 63000 inertia value and 4 has 53000, i.e. by an increase of 2 clusters, inertia has reduced by 10000 which is a significant amount as compared to when the cluster number is changed from 6 to 8 (reduced by 4000 approx).

Hence, we can choose 6 clusters to represent the data such that the inertia is minimum and this helps in defining the clusters well.

In [81]:

```
clusters, score, inertia = computeKMeans(x_scaled, n=6, max_iter=400)
kmeans = pd.DataFrame(clusters)
x_kmeans = pd.DataFrame(x_scaled).copy()
x_kmeans['kmeans clusters'] = kmeans
inertia
```

Out[81]:

```
47395.50147426181
```

In [83]:

```
x_kmeans.head()
```

Out[83]:

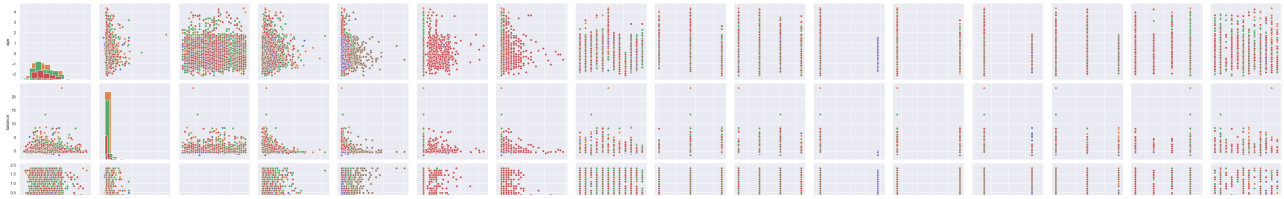
	age	balance	day	duration	campaign	pdays	previous	job	marital	education	default	housing	loan
0	-1.056270	0.121072	0.374052	-0.711861	-0.576829	-0.407218	-0.320413	1.716804	-0.246429	-1.644755	-0.130759	-1.142051	-0.424756
1	-0.772583	1.118644	-0.596026	-0.169194	-0.576829	2.989044	2.041734	0.795246	-0.246429	-0.309038	-0.130759	0.875617	2.354292
2	-0.583458	-0.024144	0.010273	-0.303898	-0.576829	2.899143	0.270124	-0.126313	1.421396	1.026680	-0.130759	0.875617	-0.424756
3	-1.056270	0.017726	-1.566105	-0.250017	0.387967	-0.407218	-0.320413	-0.126313	-0.246429	1.026680	-0.130759	0.875617	2.354292
4	1.686036	-0.472753	-1.323585	-0.146102	-0.576829	-0.407218	-0.320413	-1.047871	-0.246429	-0.309038	-0.130759	0.875617	-0.424756

Visualization of features with respect to the clusters

After the k-means clustering is performed, we add an additional column to the x_scaled dataframe such that the column values represent the respective clusters to which the given row of data belongs to. This will help us in plotting the datapoints on an SPLOM. The following visualization helps to see the distribution of data points with respect to one feature vs the other along with the clusters defined by the color of the points. Each subplot represents distribution across two features plotted using x and y axis.

In [84]:

```
import seaborn as sns
sns.set()
sns.pairplot(x_kmeans,hue='kmeans clusters',diag_kind='hist', height = 3.0)
plt.show()
```

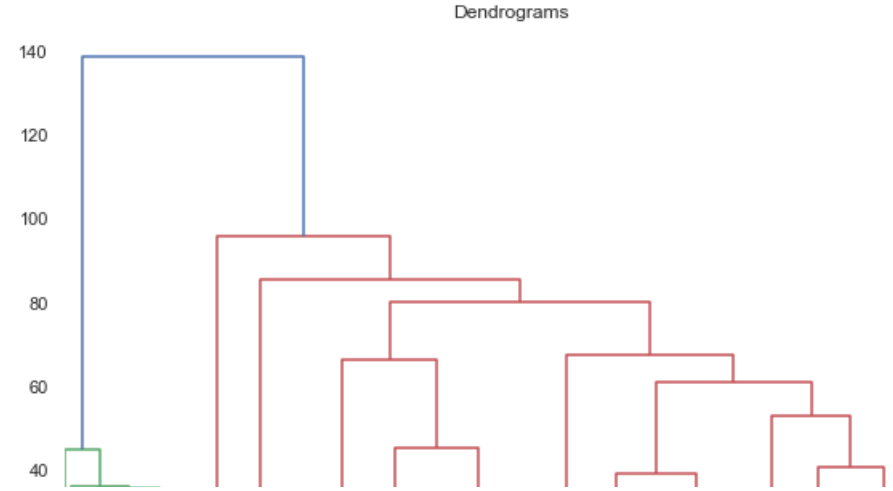


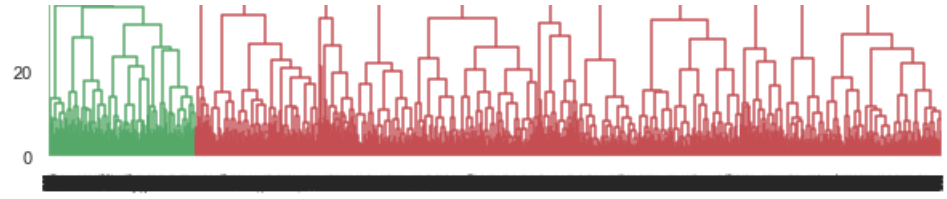
Hirarchical clustering

Lets plot the dendogram structure of the x_scaled dataset using scipy (https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html). The y-axis represents the Euclidean distance and the x-axis are the data points (observations). First we determine the linkages between data points and then plot the dendogram

In [86]:

```
import scipy.cluster.hierarchy as h
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
sh = h.linkage(x_scaled, method='ward')
dendo = h.dendrogram(sh)
```





In [87]:

```
sh
```

Out[87]:

```
array([[6.11000000e+02, 6.15000000e+02, 1.08764549e-01, 2.00000000e+00],
       [1.96600000e+03, 2.15300000e+03, 1.39148437e-01, 2.00000000e+00],
       [2.24300000e+03, 3.93800000e+03, 1.45412665e-01, 2.00000000e+00],
       ...,
       [9.02600000e+03, 9.03700000e+03, 8.52467978e+01, 3.69400000e+03],
       [8.98200000e+03, 9.03800000e+03, 9.55036043e+01, 3.77000000e+03],
       [9.03100000e+03, 9.03900000e+03, 1.38715692e+02, 4.52100000e+03]])
```

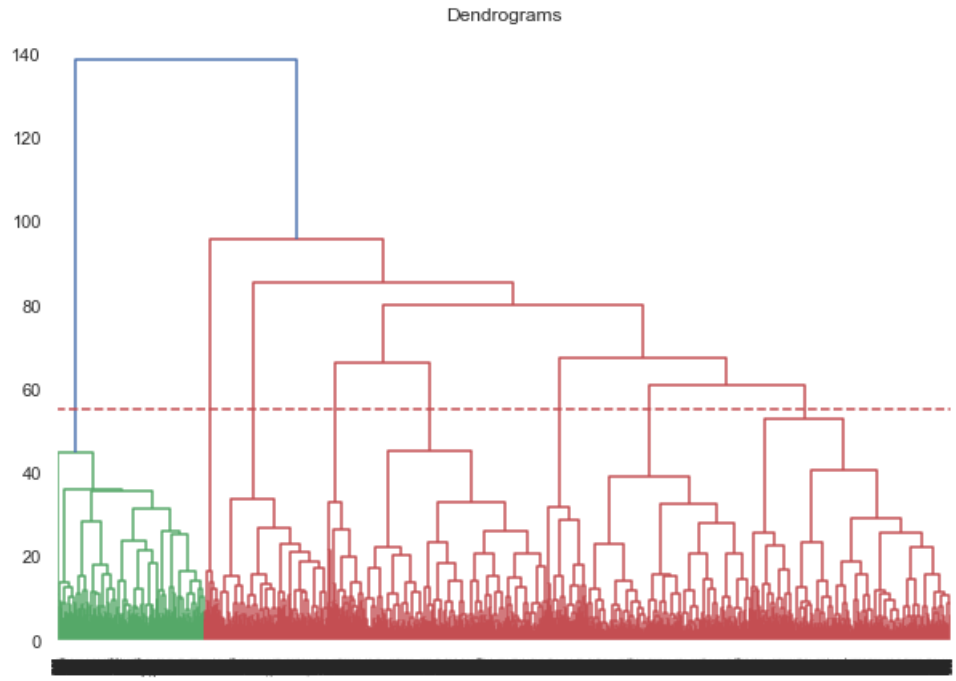
Now that we have visualized the dendogram, we can choose the threshold at which we would like to cut through the dendogram such that the vertical linkages are broken and the dendogram is divided into different clusters. By visual cue, I decide upon 55 on y-axis a good measure to divide in clusters. Hence, the number of clusters will be 8.

In [88]:

```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dendo = h.dendrogram(sh)
plt.axhline(y=55, color='r', linestyle='--')
```

Out[88]:

```
<matplotlib.lines.Line2D at 0x1cea3089ac8>
```



After visualizing the dendogram and choosing clusters we can perform agglomerative clustering which is one type of hirarchical clustering. The same can be done using AgglomerativeClustering library provided by sklearn.

In [89]:

```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 8, affinity = 'euclidean', linkage ='ward')
clusters_hc = hc.fit_predict(x_scaled)
```

```
clusters_hc
```

Out[89]:

```
array([1, 4, 4, ..., 6, 4, 4], dtype=int64)
```

In [90]:

```
hc_x = pd.DataFrame(x_scaled)
hc_x['hc_cluster'] = pd.DataFrame(clusters_hc).copy()
```

In [92]:

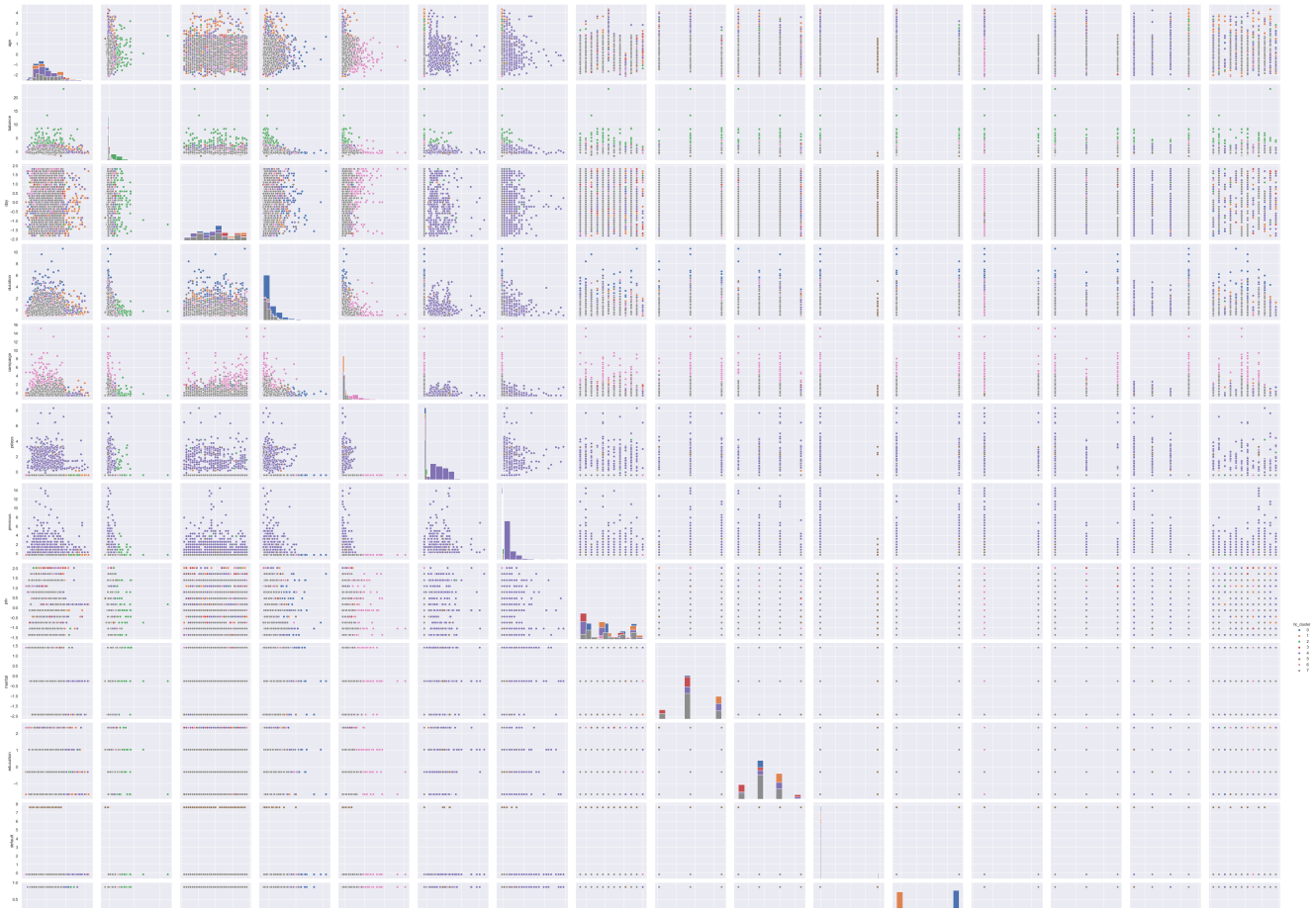
```
hc_x.head()
```

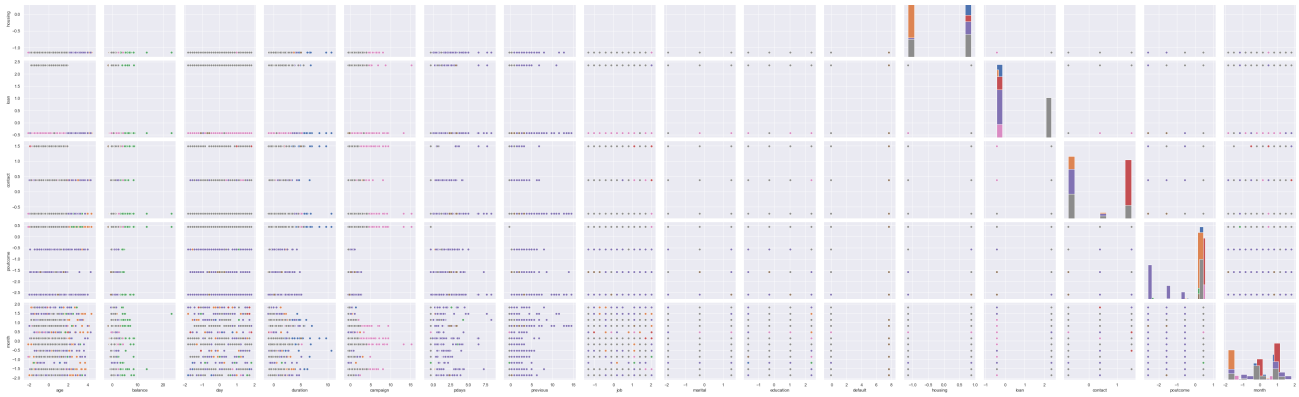
Out[92]:

	age	balance	day	duration	campaign	pdays	previous	job	marital	education	default	housing	loan
0	-1.056270	0.121072	0.374052	-0.711861	-0.576829	0.407218	-0.320413	1.716804	-0.246429	-1.644755	-0.130759	-1.142051	-0.424756
1	-0.772583	1.118644	-0.596026	-0.169194	-0.576829	2.989044	2.041734	0.795246	-0.246429	-0.309038	-0.130759	0.875617	2.354292
2	-0.583458	-0.024144	0.010273	-0.303898	-0.576829	2.899143	0.270124	-0.126313	1.421396	1.026680	-0.130759	0.875617	-0.424756
3	-1.056270	0.017726	-1.566105	-0.250017	0.387967	0.407218	-0.320413	-0.126313	-0.246429	1.026680	-0.130759	0.875617	2.354292
4	1.686036	-0.472753	-1.323585	-0.146102	-0.576829	0.407218	-0.320413	-1.047871	-0.246429	-0.309038	-0.130759	0.875617	-0.424756

In [93]:

```
import seaborn as sns
sns.set()
sns.pairplot(hc_x,hue='hc_cluster',diag_kind='hist', height = 3.0)
plt.show()
```





Inference and Comparision of K-means, PCA and Hirarchical clustering

PCA looks to find a low-dimensional representation of the observations that explain a good fraction of the variance whereas Clustering looks to find homogeneous subgroups among the observations.

The PCA algorithm is best suited for dimensionality reduction of a dataset before performing any supervised machine learning algorithms for high dimensional dataset. However, as we can see in the elbow or bar plots presented for PCA that it helps in determinig the components that best represent the data and yet can not preserve 100% of variance or information in the data if choosen even 1 less than the maximum number of components available. That is to say, we reduce the dimensions by compromising on some information in the data. PCA works by taking projections of data points on a given component and then it either measures the distances from the data to the line and tries to find the line that minimizes those distances or it can try to find the line that maximizes the distances from the projected points to the origin.

K-means on the other hand uses data points to centroid distance (Euclidean or any distance) measure to evaluate or group datasets into different clusters. This is not specifically losing up on data unlike PCA which projects data points onto different components (eigen vectors) to summarize the variablilty of data in the best posisble way. K-means focuses on cluster analysis rather than component analysis which results in grouping of data into relevant clusters. Choice of clusters depends on the factor such as how inter cluster distances are maximized and intra cluster distances are minimized.

Hirarchical clustering could be assumed as a better version than k-means where we do not have to worry on choosing the clusters before fitting the data to the model. Once the data is fit using hirarchical clustering we can use the threshold on Euclidean distance such that it can divide the complete tree or dendogram into individual clusters

Based on the analysis of the visualizations presented in this document, clustering using hirarchical is more intuitive as compared to the k-means. While I agree, k-means suits best for few scenarios where the data is quite dispersed as in to identify well defined clusters by a simple visualization of the spread. But, in cases where the data is high dimensional and is extremely overlapping i.e. a simple scatter plot visualization will never help to decide the clusters rather techniques like elbow method for evaluating inertia are choosen, I woud prefer hirarchical clustering where the visual cue is more easy to understand via dendograms.