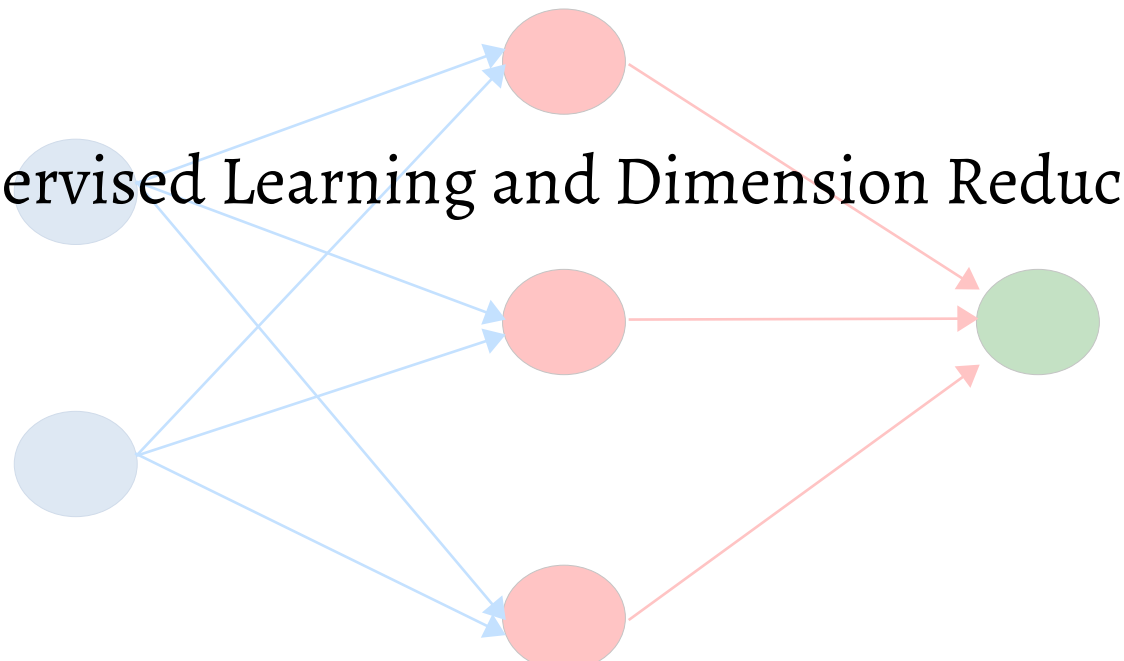
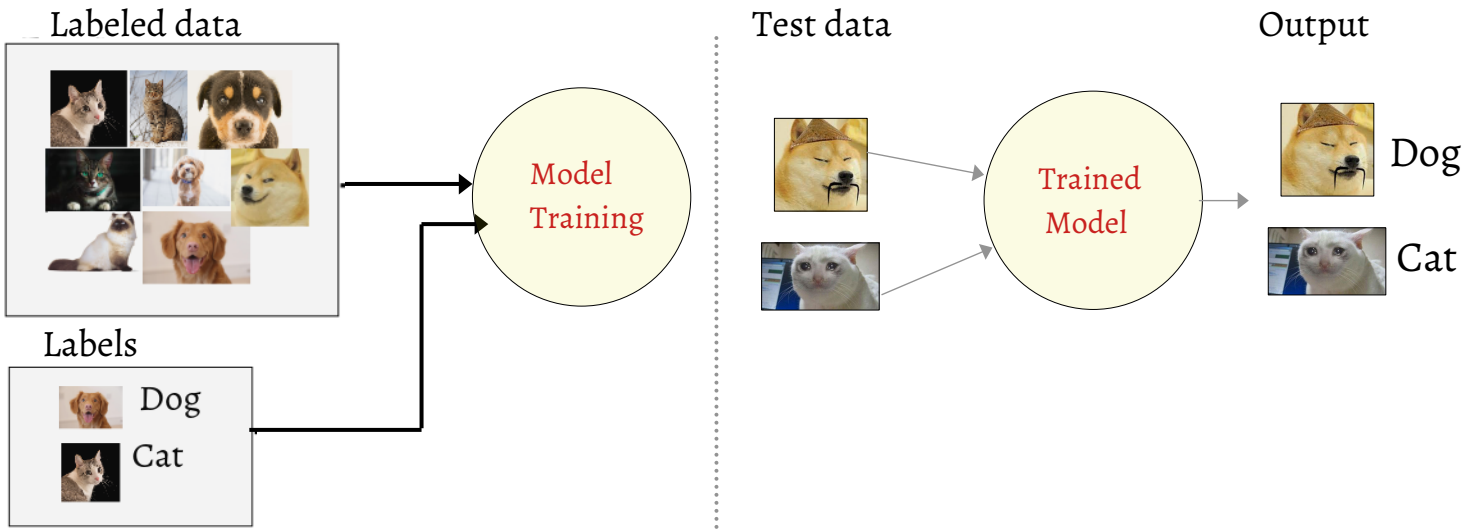
PH6232: Machine Learning for Physics applications



Unsupervised Learning and Dimension Reduction

March 17th, 2023

Supervised Learning



- Supervised learning: Training on labeled data
- Network uses labels to assess the training and feedbacks are propagated

But the labels(or true identification) may not be available in many problems

So far in the course...

Unsupervised Learning

- Learns from dataset without labels
- It can find patterns and structure in the dataset

Let the algorithm learns whatever is meaningful

These are different kinds of table and
chairs you spotted at IISERP Physics dept!

Unsupervised Learning

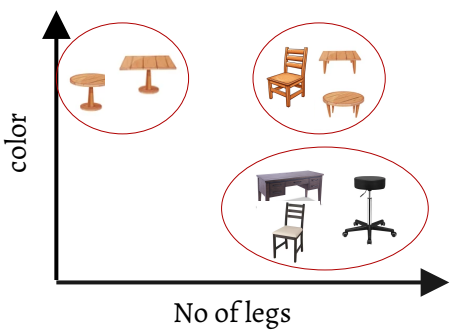
- Learns from dataset without labels
- It can find patterns and structure in the dataset

Let the algorithm learns whatever is meaningful

These are different kinds of table and
chairs you spotted at IISERP Physics dept!

1) Clustering analysis

- grouping based on similarities in features



- Kmeans
- DBSCAN, OPTICS and many more

clustering sklearn_clustering

Unsupervised Learning

1) Clustering analysis

- grouping based on similarities in features
- Kmeans, DBSCAN, OPTICS and many more algorithms

2) Data Compression

- you are a smart person and you note down many features of the chair and tables

- 1. No of legs
  2. Connection between legs
  3. Height from ground
  4. Color
  5. Back rest
  6. Drawers
  7. Geometry of the flat surface
  8. Faculty or student or admin or library

→

- One can perform clustering analysis in this multidimensional space
- But many features are abundant or not really matters to find the intrinsic patters of the dataset
- All features or dimensions are not necessary. They don't bring any new substantial information
  - we can either remove them manually or compress them in low dimensional space
- Dimension Reduction algorithms:
  - PCA, TSNE, UMAP etc
  - Easy to visualize, they can form cluster as well!
  - Saves computing power, storage

Unsupervised Learning

1) Clustering analysis

- grouping based on similarities in features
- Kmeans, DBSCAN, OPTICS and many more algorithms

1. Recommending system(such as what Post should come in your insta, fb feed, Netflix movie suggestion etc)

2. Cluster energetic deposits or hits of particles

2) Data Compression

- High dimensional space to low dimensional space (Latent space)
- PCA, TSNE, UMAP

3) Anomaly or Outlier Detection

- Detect any data points that is different from the bulk of the dataset
- Many algorithms such as Autoencoders, KNN etc
- Autoencoders will be discussed next week

1. Detecting fraud bank transaction

2. Discovering new physics beyond Standard Model

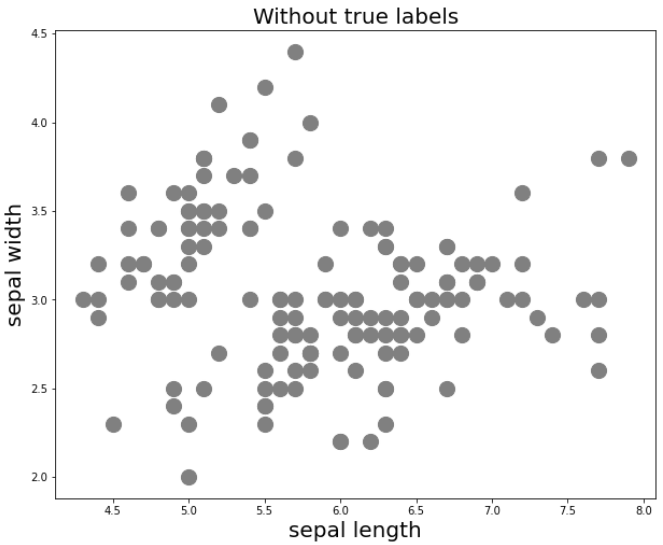
Clustering Analysis

- Standard dataset from sklearn library: Iris Dataset
- Three species of flower: Setosa, virginica and versicolor
- Four features: Sepal width, sepal length, petal width and petal length

```
df_irisdata.head(5)
```

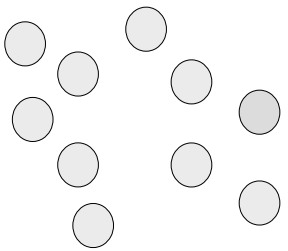
	sepal_length	sepal_width	petal_length	type
36	5.5	3.5	0.2	0
34	4.9	3.1	0.2	0
51	6.4	3.2	1.5	1
104	6.5	3.0	2.2	2
107	7.3	2.9	1.8	2

Let's assume that we don't know the label



How many groups we can see?

KMeans algorithm



1. choose the number of cluster k

- lets pick 2

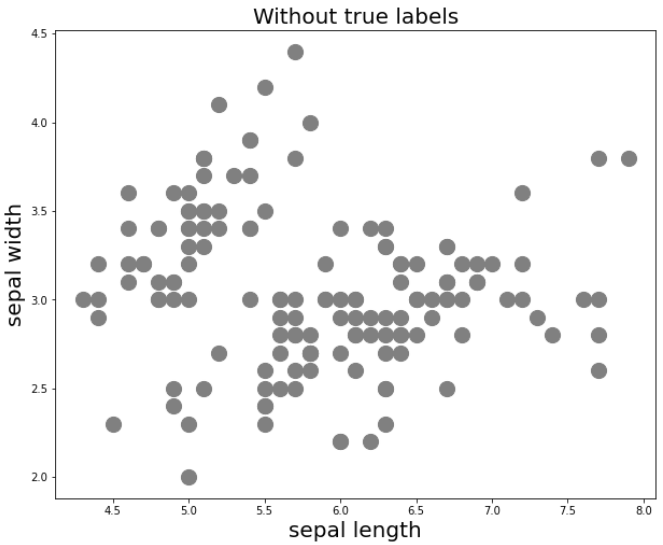
Clustering Analysis

- Standard dataset from sklearn library: Iris Dataset
- Three species of flower: Setosa, virginica and versicolor
- Four features: Sepal width, sepal length, petal width and petal length

```
df_irisdata.head(5)
```

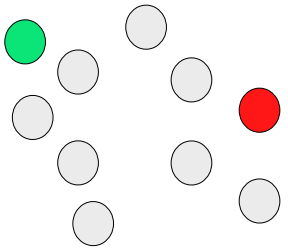
	sepal_length	sepal_width	petal_length	type
36	5.5	3.5	0.2	0
34	4.9	3.1	0.2	0
51	6.4	3.2	1.5	1
104	6.5	3.0	2.2	2
107	7.3	2.9	1.8	2

Let's assume that we don't know the label



How many groups we can see?

KMeans algorithm



1. choose the number of cluster k

2.Take 'k' random point as a centroid to starts with

- calculate distance of each point from these centroids

- assign them to the closest centroid

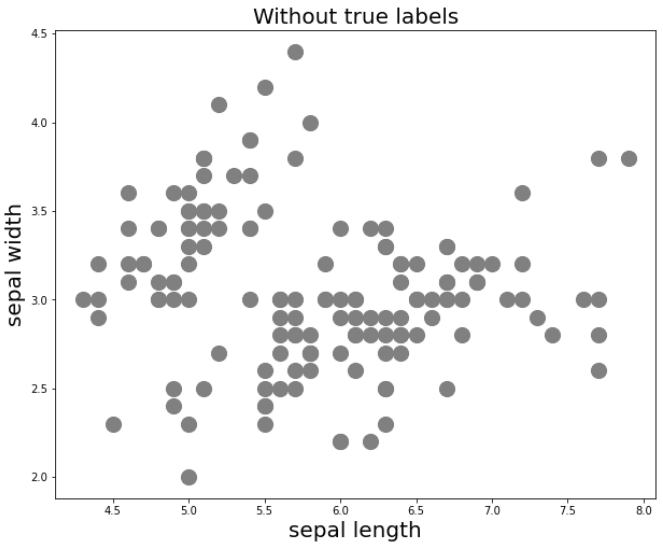
Clustering Analysis

- Standard dataset from sklearn library: Iris Dataset
- Three species of flower: Setosa, virginica and versicolor
- Four features: Sepal width, sepal length, petal width and petal length

```
df_irisdata.head(5)
```

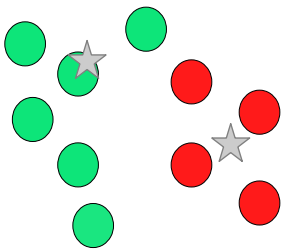
	sepal_length	sepal_width	petal_length	type
36	5.5	3.5	0.2	0
34	4.9	3.1	0.2	0
51	6.4	3.2	1.5	1
104	6.5	3.0	2.2	2
107	7.3	2.9	1.8	2

Let's assume that we don't know the label



How many groups we can see?

KMeans algorithm



1. choose the number of cluster k

2.Take 'k' random point as a centroid to starts with

3. Calcuate distance of all points from the centroid and assign those points to the closest centroid

Repeat

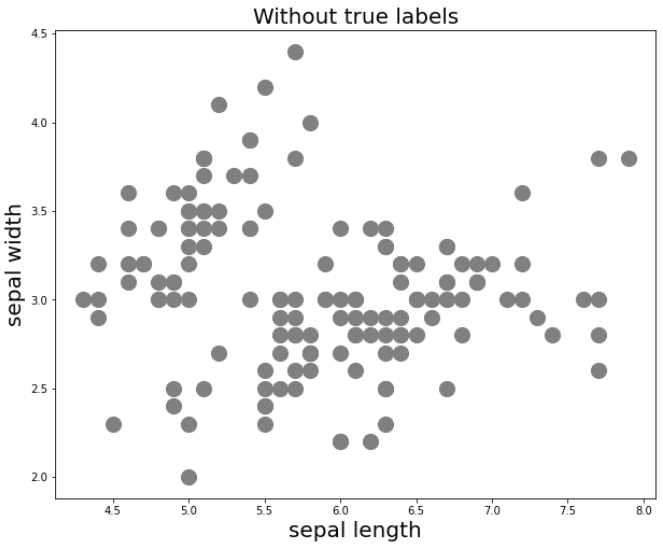
Clustering Analysis

- Standard dataset from sklearn library: Iris Dataset
- Three species of flower: Setosa, virginica and versicolor
- Four features: Sepal width, sepal length, petal width and petal length

```
df_irisdata.head(5)
```

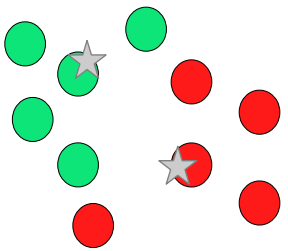
	sepal_length	sepal_width	petal_length	type
36	5.5	3.5	0.2	0
34	4.9	3.1	0.2	0
51	6.4	3.2	1.5	1
104	6.5	3.0	2.2	2
107	7.3	2.9	1.8	2

Let's assume that we don't know the label



How many groups we can see?

KMeans algorithm



1. choose the number of cluster k

2.Take 'k' random point as a centroid to starts with

3. Calcuate distance of all points from the centroid and assign those points to the closest centroid

Repeat

Stop when,

- centroids of new clusters do not change
- Points remain in the same cluster
- Reaches maximum number of iterations

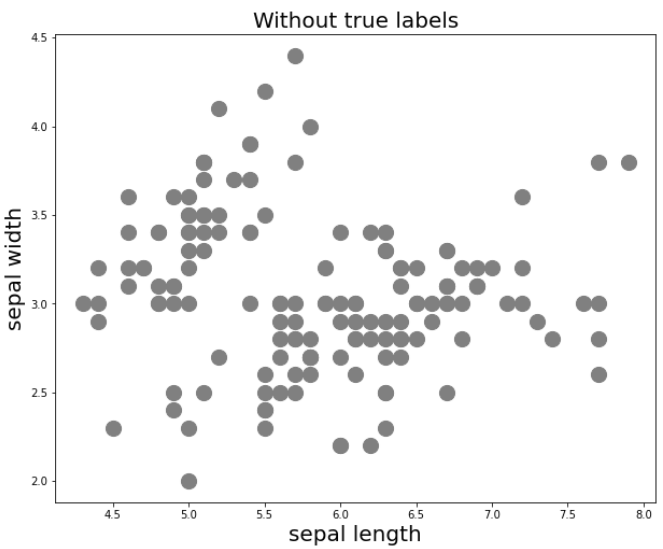
Clustering Analysis

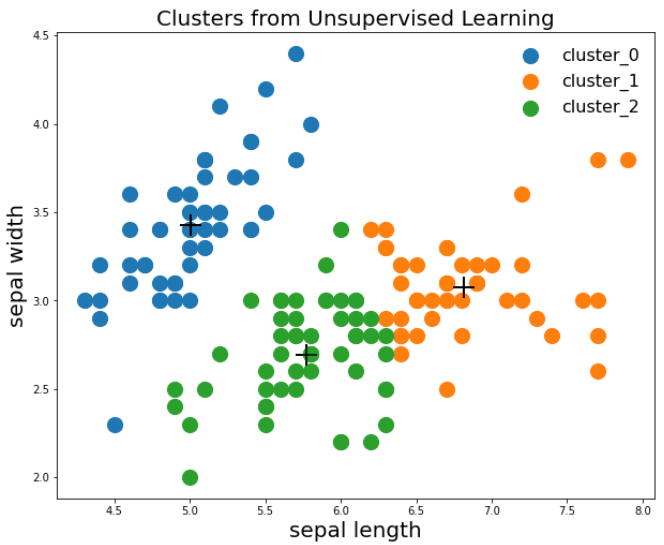
- Standard dataset from sklearn library: Iris Dataset
- Three species of flower: Setosa, virginica and versicolor
- Four features: Sepal width, sepal length, petal width and petal length

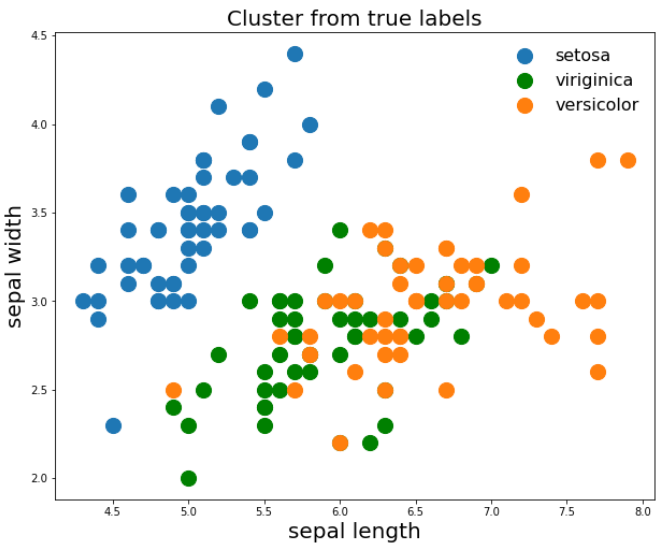
```
df_irisdata.head(5)
```

	sepal_length	sepal_width	petal_length	type
36	5.5	3.5	0.2	0
34	4.9	3.1	0.2	0
51	6.4	3.2	1.5	1
104	6.5	3.0	2.2	2
107	7.3	2.9	1.8	2

Result on the Iris Dataset







Unsupervised

If we knew the true labels!

Disadvantage: poor performance on closely spaced points(where density is large)

Clustering Analysis Code

from sklearn.cluster import KMeans

In [85]:
```
kmeans = KMeans(n_clusters=3, init='k-means++')

# fitting the k means algorithm
kmeans.fit(df_unsupervised)
```

Out[85]:
```
KMeans(n_clusters=3)
```

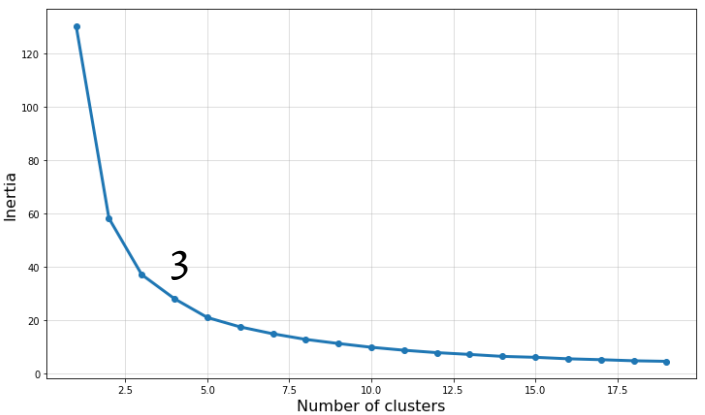
In [86]:
```
cluster = kmeans.predict(df_unsupervised)

```

In [87]:
```
df_unsupervised['cluster_label']=cluster

#no of unique cluster
clusters = np.unique(cluster)
centers=kmeans.cluster_centers_
df_unsupervised.head()
```

This Easy!

But wait, how do you choose 3 cluster to begin with?

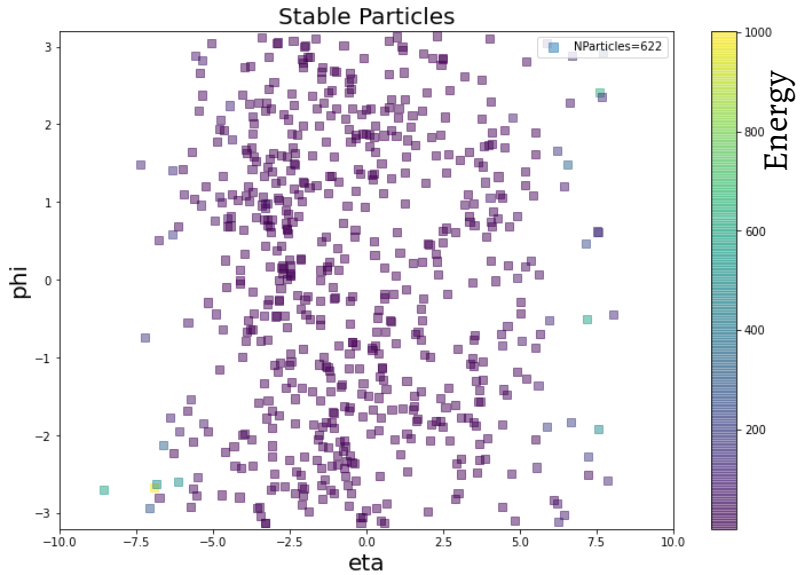
Elbow method!



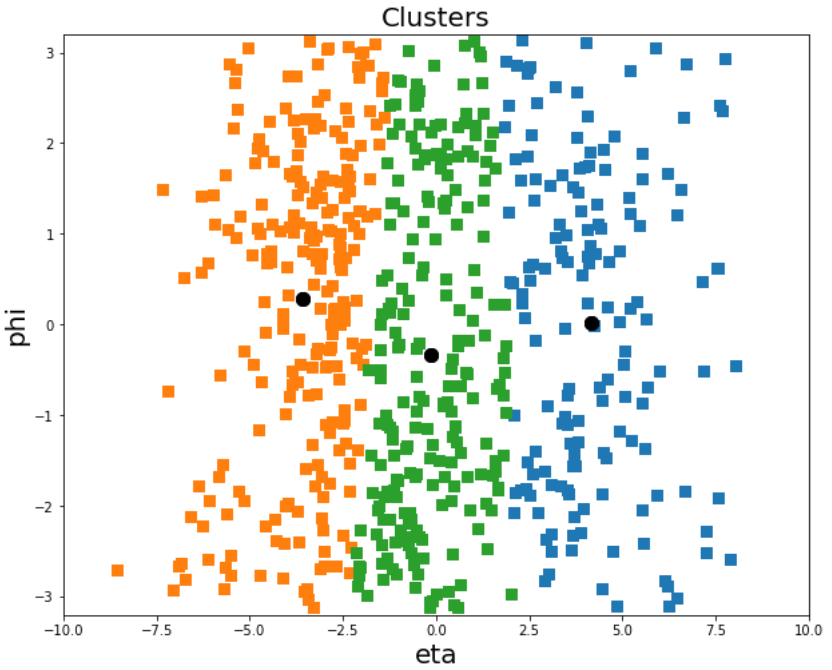
- Run for different no of cluster
- Metric is Total Intertia
- Interia: Sum of distances of all the points from centroid of that cluster
- Optimum no of cluster can be chosen where the elbow is

Applying on the HEP Problem

- Dataset contains no of particles on the eta-phi plane



Applying KMean on this problem to find clusters

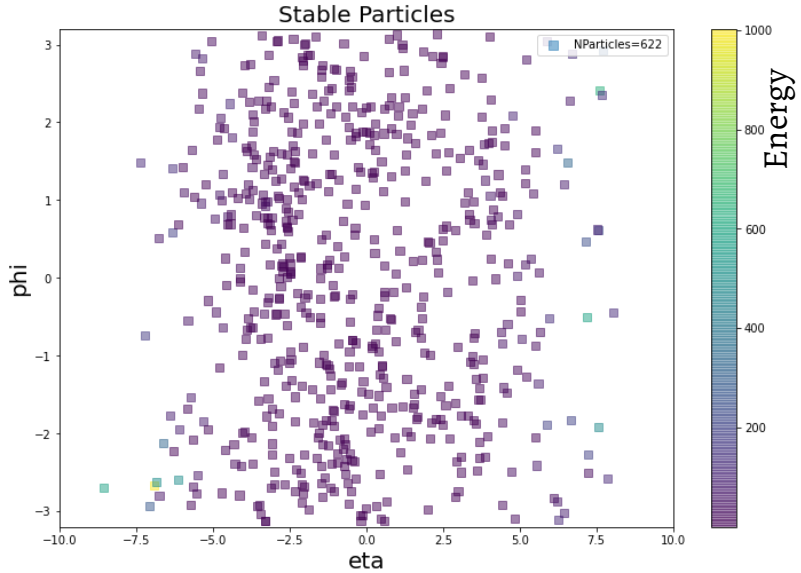


- Fails to find sensible clusters
- Centroid based algorithm:

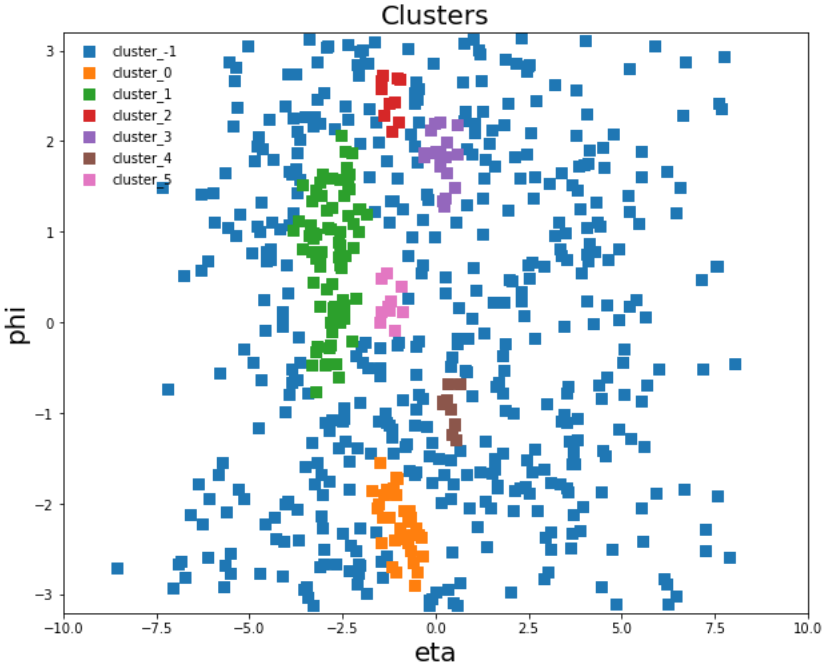
  - works really bad when densisty of points are varying
  - fail in creating clusters of arbitrary shapes

Applying on the HEP Problem

- Dataset contains no of particles on the eta-phi plane



Applying DBSCAN on this problem to find clusters



- This looks reasonable!

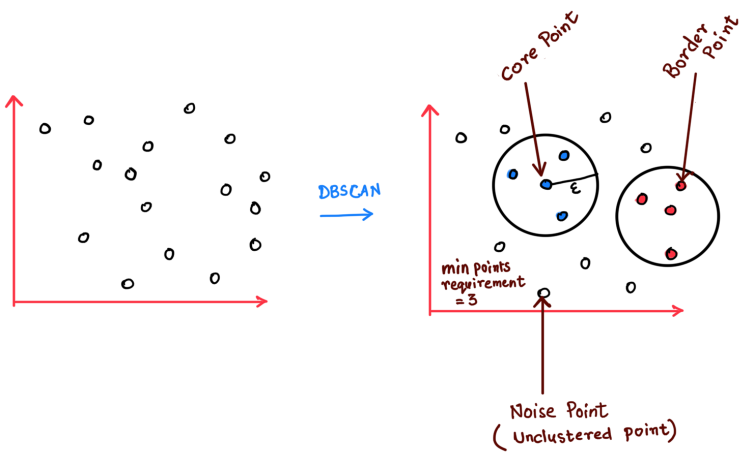
So, what is this algorithm?

DBSCAN algorithm

- Density based algorithm (KMeans is centroid based)
- DBSCAN: Density-based spatial clustering of applications with noise (paper_link)
- Based on assumption: clusters are dense regions in space separated by regions of lower density
- It depends on two parameters,

Epsilon(ε)= how close points should be to be considered a part of a cluster

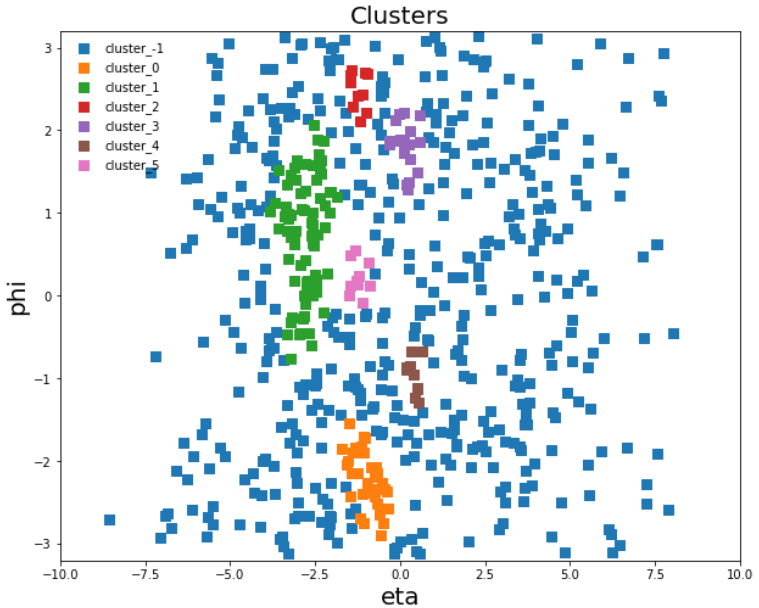
No of minimum points: min points required to form a cluster



- It scans over each point once
- A point within it's radius = ε,
  - min no of points: core point
  - at least one core points but <min points: Border point
  - <min no of points: Noise

visualizing-dbscan-clustering

Applying on the HEP Problem



ε=0.4
Min points=10

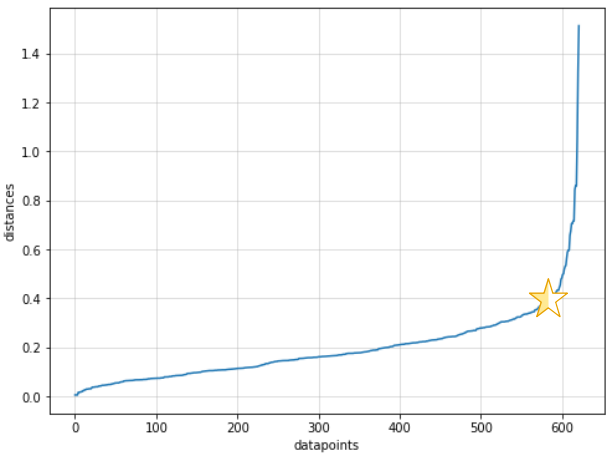
Code Block

```
In [17]:  1 from sklearn.cluster import DBSCAN

In [ ]:   1

In [18]:  1 model = DBSCAN(eps=0.40, min_samples=10)
          2
          3 # Find the clusters in the smaller dataframe
          4 yhat = model.fit_predict(df_etaphi)
          5
          6 clusters = np.unique(yhat)
```

Change the parameters and check how many cluster you are getting

Optimal Value of **ε**

- Calculate nearest neighbour distances between points
- Scan over all the points
- Choose the distances where distance between two neighbours shoots up(max curvaure)



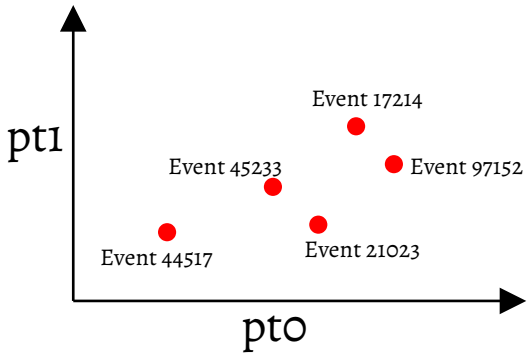
Domain knowledge is important to decide what is best!

Dimensionality Reduction

- Let's understand the multidimensionality in terms of WZ vs ZZ problem

```
WZdf.sample(5)
```

	Pt0	Pt1	Pt2	NBJet	Met	MaxDphi_LMet	MaxDphi_LL	MinDphi_LL	LLPairPt	Mt0	Mt1	Mt2
17214	111.297340	102.368149	47.637390	0.0	154.852325	1.836663	2.034217	0.733969	199.464005	26.927629	68.607086	122.485924
44517	37.767117	28.361715	24.503748	0.0	26.111670	2.606581	2.687385	0.069427	19.067734	36.326527	52.360256	71.318069
45233	71.864754	46.848637	13.040688	0.0	141.715164	2.608019	1.346833	0.209067	100.920181	238.325439	248.843155	176.257858
97152	113.258553	53.547009	24.209955	0.0	136.390152	2.920245	1.979958	0.627200	159.735931	293.612030	292.050507	177.513947
21023	102.099800	30.666338	25.705837	0.0	137.357666	2.977186	2.134700	0.231808	96.556755	229.760086	256.071320	112.581902



- If we only plot each event in pt0-pt1 plane, we can identify each event by two numbers (pt0,pt1) : Two Dimensional (2D)

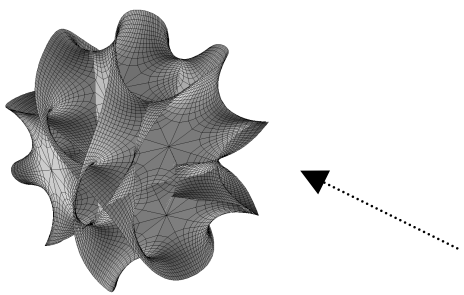
Dimensionality Reduction

- Let's understand the multidimensionality in terms of WZ vs ZZ problem

```
WZdf.sample(5)
```

	Pt0	Pt1	Pt2	NBJet	Met	MaxDphi_LMet	MaxDphi_LL	MinDphi_LL	LLPairPt	Mt0	Mt1	Mt2
17214	111.297340	102.368149	47.637390	0.0	154.852325	1.836663	2.034217	0.733969	199.464005	26.927629	68.607086	122.485924
44517	37.767117	28.361715	24.503748	0.0	26.111670	2.606581	2.687385	0.069427	19.067734	36.326527	52.360256	71.318069
45233	71.864754	46.848637	13.040688	0.0	141.715164	2.608019	1.346833	0.209067	100.920181	238.325439	248.843155	176.257858
97152	113.258553	53.547009	24.209955	0.0	136.390152	2.920245	1.979958	0.627200	159.735931	293.612030	292.050507	177.513947
21023	102.099800	30.666338	25.705837	0.0	137.357666	2.977186	2.134700	0.231808	96.556755	229.760086	256.071320	112.581902

- 12 variables to describe an WZ event



- If we only plot each event in pt0-pt1 plane, we can identify each event by two numbers (pt0,pt1) : Two Dimensional (2D)
- I can also imagine a 12 Dimensional hyperspace where each WZ event can be identified by 12 numbers: (pt0,pt1.....Mt2)
  - hard to imagine even!

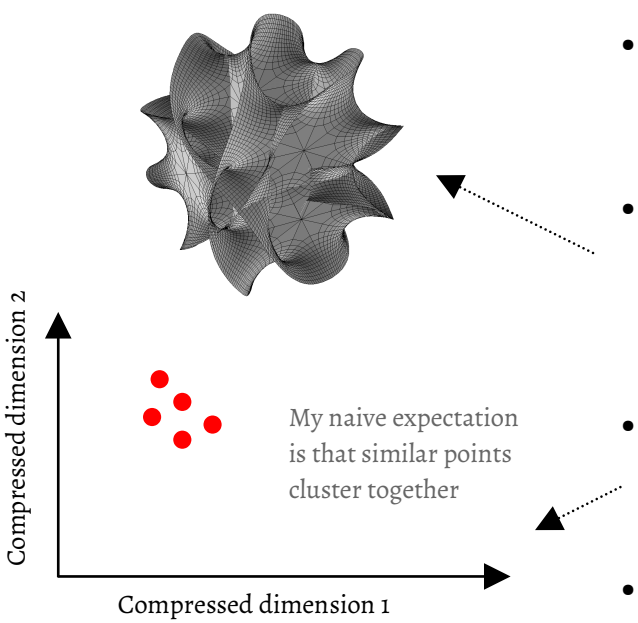
Dimensionality Reduction

- Let's understand the multidimensionality in terms of WZ vs ZZ problem

```
WZdf.sample(5)
```

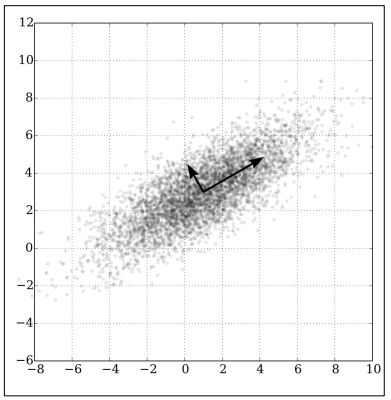
	Pt0	Pt1	Pt2	NBJet	Met	MaxDphi_LMet	MaxDphi_LL	MinDphi_LL	LLPairPt	Mt0	Mt1	Mt2
17214	111.297340	102.368149	47.637390	0.0	154.852325	1.836663	2.034217	0.733969	199.464005	26.927629	68.607086	122.485924
44517	37.767117	28.361715	24.503748	0.0	26.111670	2.606581	2.687385	0.069427	19.067734	36.326527	52.360256	71.318069
45233	71.864754	46.848637	13.040688	0.0	141.715164	2.608019	1.346833	0.209067	100.920181	238.325439	248.843155	176.257858
97152	113.258553	53.547009	24.209955	0.0	136.390152	2.920245	1.979958	0.627200	159.735931	293.612030	292.050507	177.513947
21023	102.099800	30.666338	25.705837	0.0	137.357666	2.977186	2.134700	0.231808	96.556755	229.760086	256.071320	112.581902

- 12 variables to describe an WZ event



- If we only plot each event in pt0-pt1 plane, we can identify each event by two numbers (pt0,pt1) : Two Dimensional (2D)
- I can also imagine a 12 Dimensional hyperspace where each WZ event can be identified by 12 numbers: (pt0,pt1.....Mt2)
  - hard to imagine even!
- We can compress the information of 12 variables in just 2 variables (with some information loss) : Dimension Reduction
- Easy to visualize and in the process we may find some patterns!
  - compressed space are called latent space

Dimensionality Reduction Algorithms

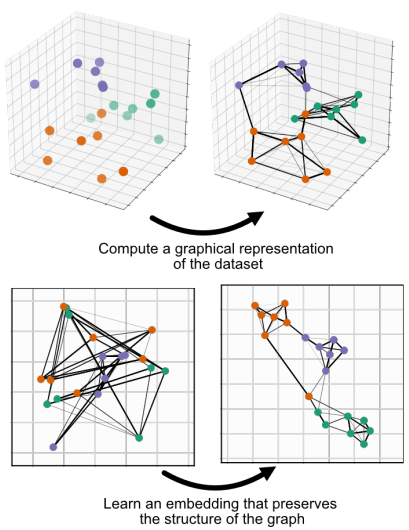


PCA

Covariance Matrix Q

$$Q = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$$

- Linear method to reduce dimension
- Use covariance matrix
- Eigenvectors of this matrix become the new basis to represent old data
- Component with largest eigenvalue => Principal component
- Keep the variance in data as much as possible in fewer components



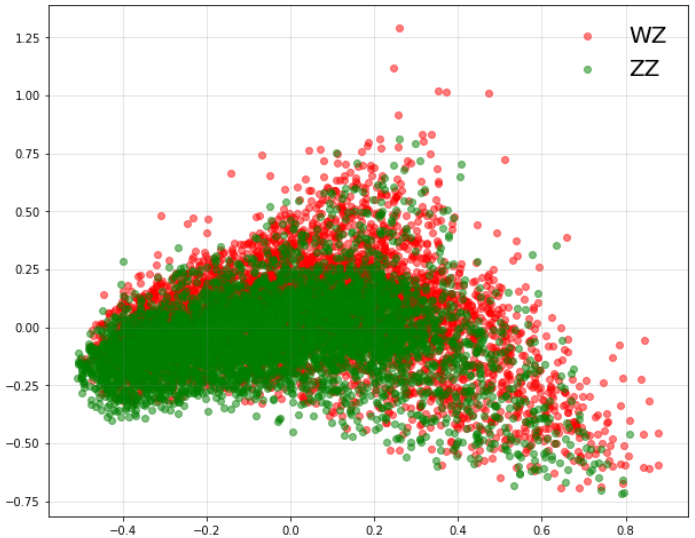
Credit: UMAP Documentation

UMAP

- Uniform Manifold Approximation & Projection
- Create high dimensional graph based on some similarity measure
- Then make a low dimensional graph that is “closer” to the high dimensional graph
- Check how “similar class” clustered in this low dimensional latent space

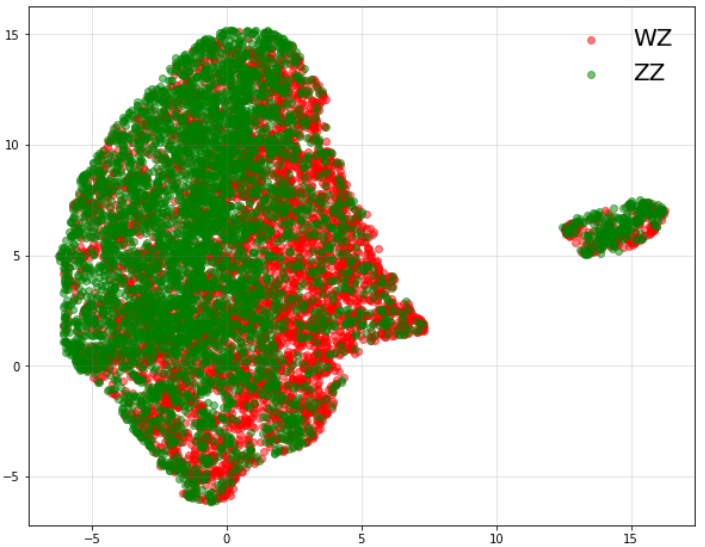
WZ and ZZ in Latent Space

PCA




```
from sklearn.decomposition import PCA

####PCA
model_pca=PCA(n_components=2)
model_pca.fit(X_train)
x_pca = model_pca.transform(X_train)

```

UMAP



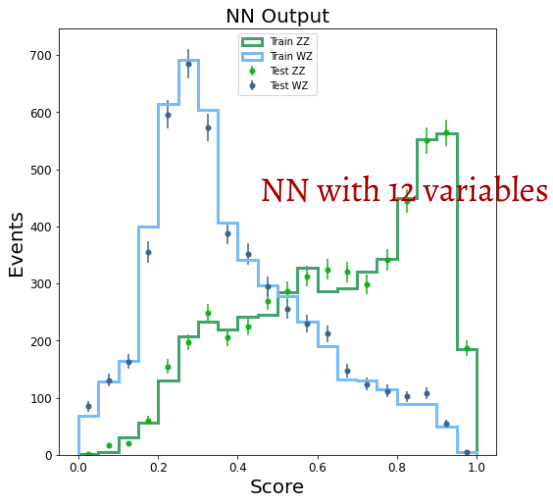

```
import umap
```

You have to install the umap package

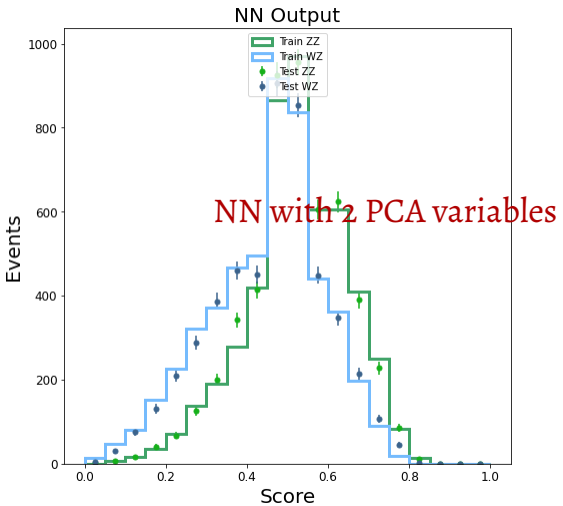
```
#model umap
model_umap=umap.UMAP(n_neighbors=20, min_dist=0.4, n_components=2,metric='euclidean')
model_umap.fit(X_train)
x_umap = model_umap.transform(X_train)

```

n_neighbours, min_dist, n_components, metric are all the knobs you can play with

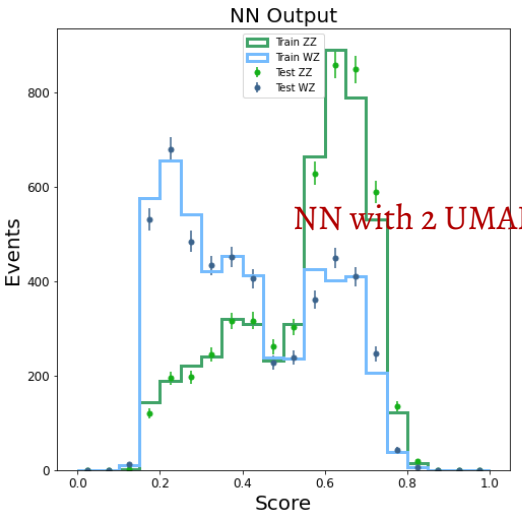
Neural Network training on latent variables



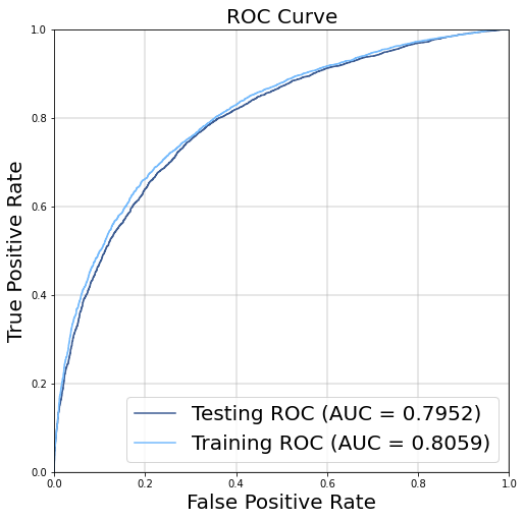
NN with 12 variables

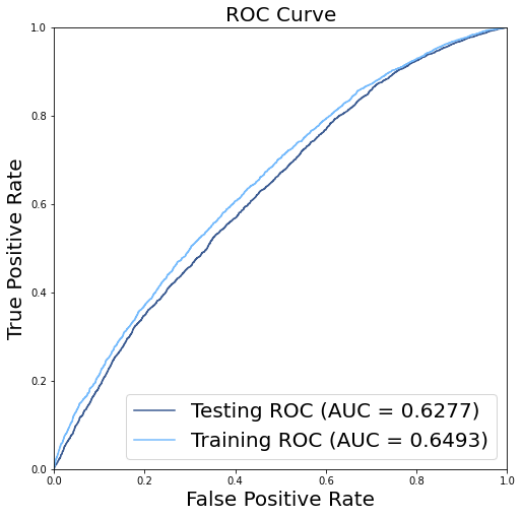


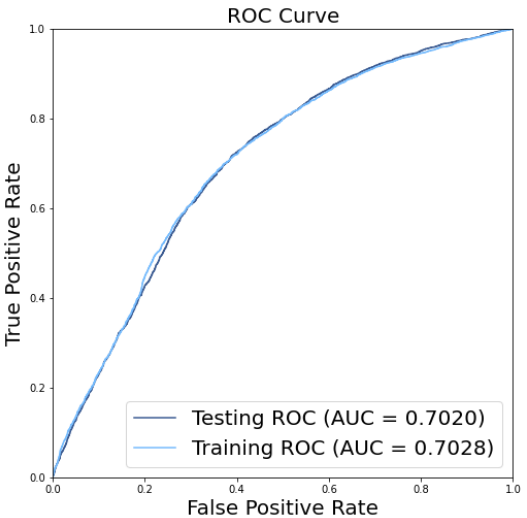
NN with 2 PCA variables



NN with 2 UMAP variables







Your Task

- Reduce 12 dimension to 3 dimension and visualize WZ , ZZ events in 3D space
- Reduce 12 dimension to 5 dimension and visualize WZ , ZZ events in 2D space of different latent variables
- Conduct a training with 5 latent variables and compare the ROC