Data Exploration

Dr Haider Raza
(Slides adapted from Dr Spyros Samothrakis)
Postdoctoral Research Fellow @IADS
University of Essex

February 18, 2019

About

- We will be discussing ways of understanding the data
  - Without assuming there is something to predict
- For example, you might want to split your wine data into different groups
  - But you have no idea which groups are out there
- You just have a description of each sample
  - For example each $< Alcohol, MalicAcid, Ash, ... >$

How often do people do it?

- Labelled data is often not enough or too little
  - ...and often requires human intervention
- We tend to group things all the time
  - Short / tall
  - Fast / slow
  - Certain types of clothing
- ...etc

Exploring data

"If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don't know how to make the cake."

– Yann LeCun (Director of AI Research, Facebook)

Data Science Operations

- Descriptive
  - "I'll create high level views of your data"
  - What is sometimes termed analytics
- Predictive
  - "I'll try to predict a possible version of your future".
  - Show me a some good customers in your database, I'll try getting you more
- Prescriptive
  - "What should I do to achieve certain results?"
  - Reinforcement Learning and Causality

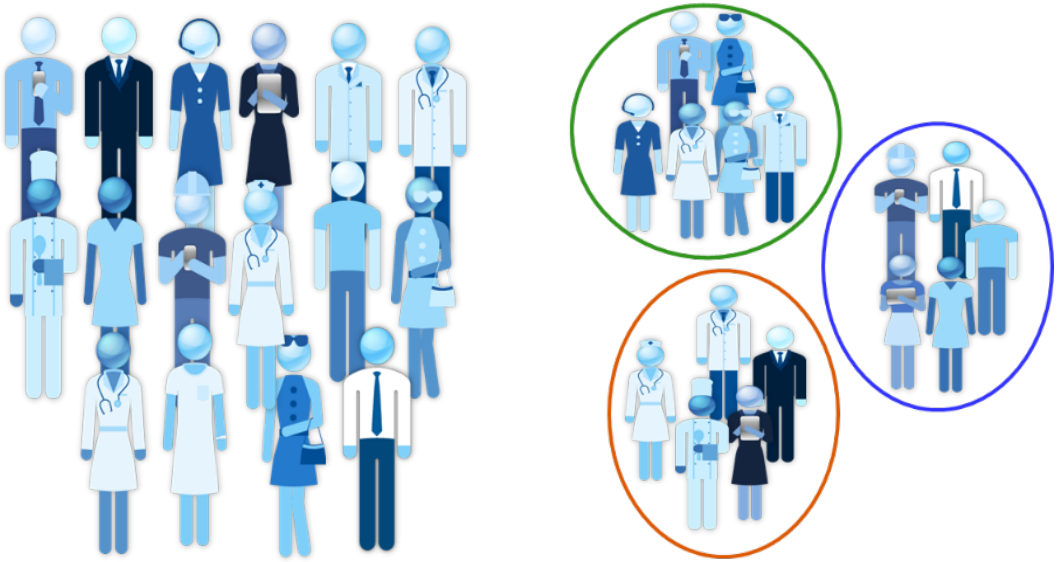
[1] Chris Wiggins. "Lectures delivered Aug 8-9, 2016 at MLSS.cc (Arequipa, Peru)".
https://www.slideshare.net/chrishwiggins/machine-learning-summer-school-2016/75

Types of learning (again)

- Supervised Learning
  - Predictions
- Reinforcement Learning
  - (very close to bandits)
- Unsupervised Learning
  - Learning about the data without any signal
  - We are not doing that well (but this might be about to change)
  - Alternatively you can try to predict all your features!

Clustering

- We would like to group our data into different groups[1]

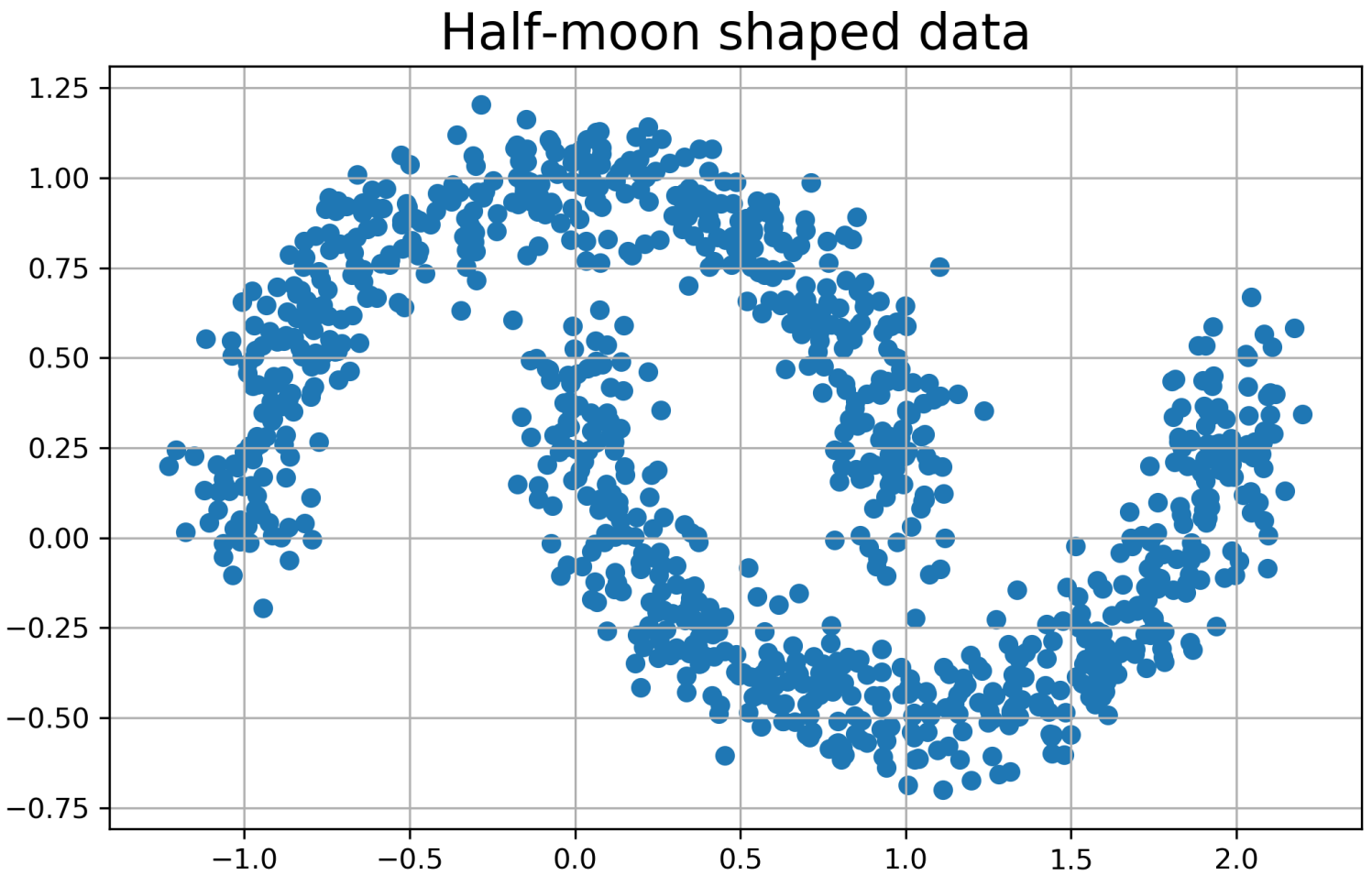


[1] https://blogs.sas.com/content/subconsciousmusings/2016/05/26/data-mining-clustering/

Code: Generate the half-moon data

```
from sklearn import datasets
n_samples = 1500 # no.of.data points

# The dataset function is avialable in sklearn package
noisy_moons,moon_labels = datasets.make_moons(n_samples=n_samples, noise=.1) # Generate Moon Toy Dataset

# Put in Array
noisy_moons=np.array(noisy_moons)

# Plot Half-moon data
plt.figure(figsize=(8,5))
plt.title("Half-moon shaped data", fontsize=18)
plt.grid(True)
plt.scatter(noisy_moons[:,0],noisy_moons[:,1])
plt.savefig('HALF_MOON.png', dpi=300)
plt.show()
```

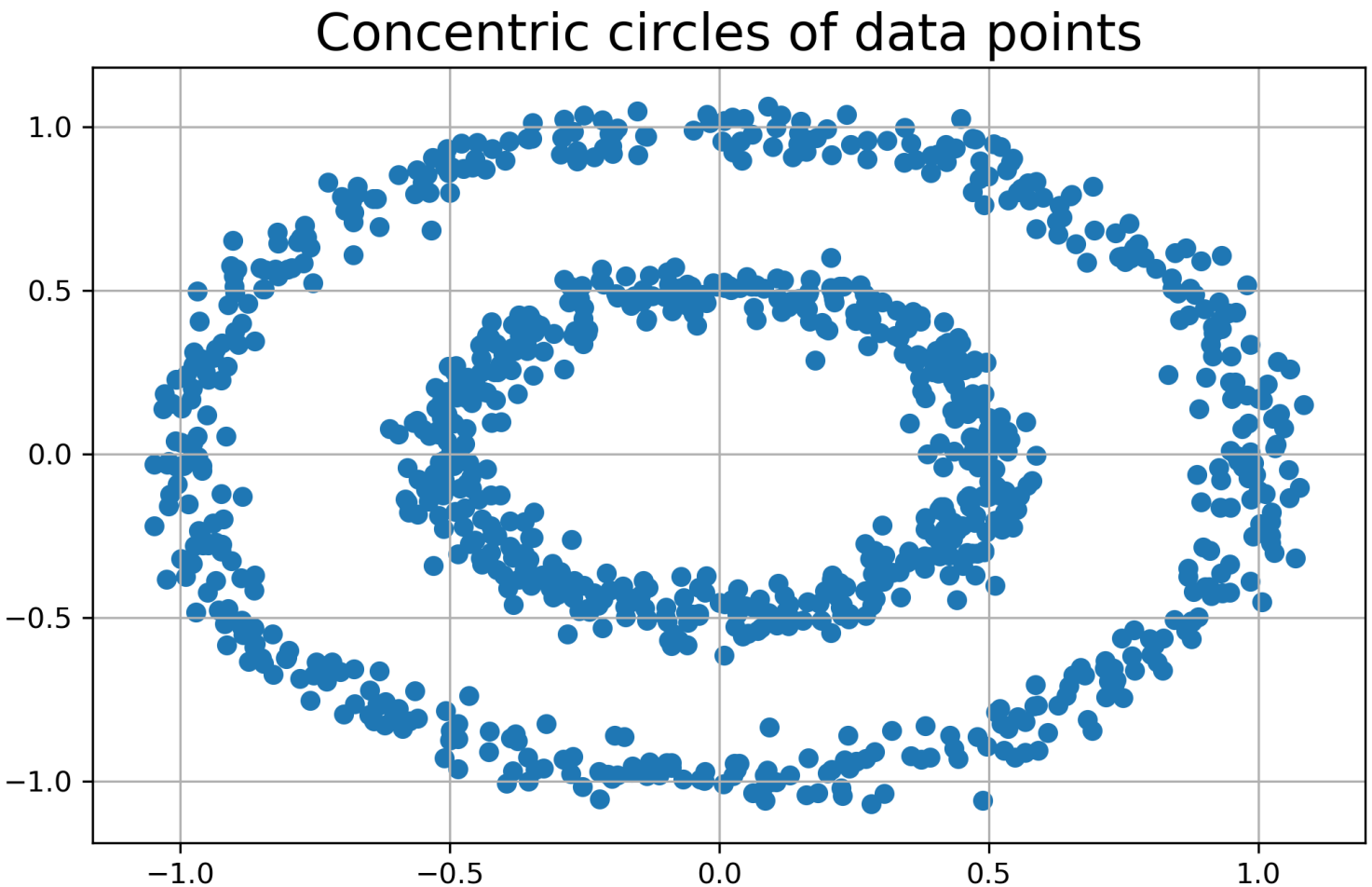
Half-Moon Data with 1500 points



Code: Generate the circle data

```
from sklearn import datasets
n_samples = 1500 # no.of.data points

# The dataset function is avialable in sklearn package
# Generate Circle Toy Dataset
noisy_circles,circle_labels = datasets.make_circles(n_samples=n_samples, factor=.5, noise=.05)

# Put in Array
noisy_circles=np.array(noisy_circles)

# Plot Circle data
plt.figure(figsize=(8,5))
plt.title("Concentric circles of data points", fontsize=18)
plt.grid(True)
plt.scatter(noisy_circles[:,0],noisy_circles[:,1])
plt.savefig('CIRCLE.png', dpi=300)
plt.show()
```

Circle Data with 1500 points

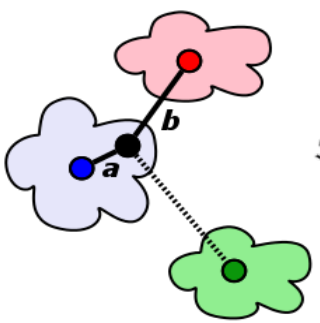


k-Means

- Possibly the most popular algorithm for clustering
- k-Means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
- Initialise with "$n\_clusters$" random "centroids"
- Iterates over two steps
  - Assign each point to one of the centroids it is closer to using euclidean distance
  - Create new centroids by defining each centroid as the average of each dimension
- Repeat
- Algorithms is unstable, different starting positions will result in different clusters

Metrics for clustering

1. Completeness: clustering must assign all of those datapoints that are members of a single class to a single cluster
2. Silhouette Coefficient:
   - +1 indicate that the sample is far away from the neighboring clusters
   - 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters
   - negative values indicate that those samples might have been assigned to the wrong cluster

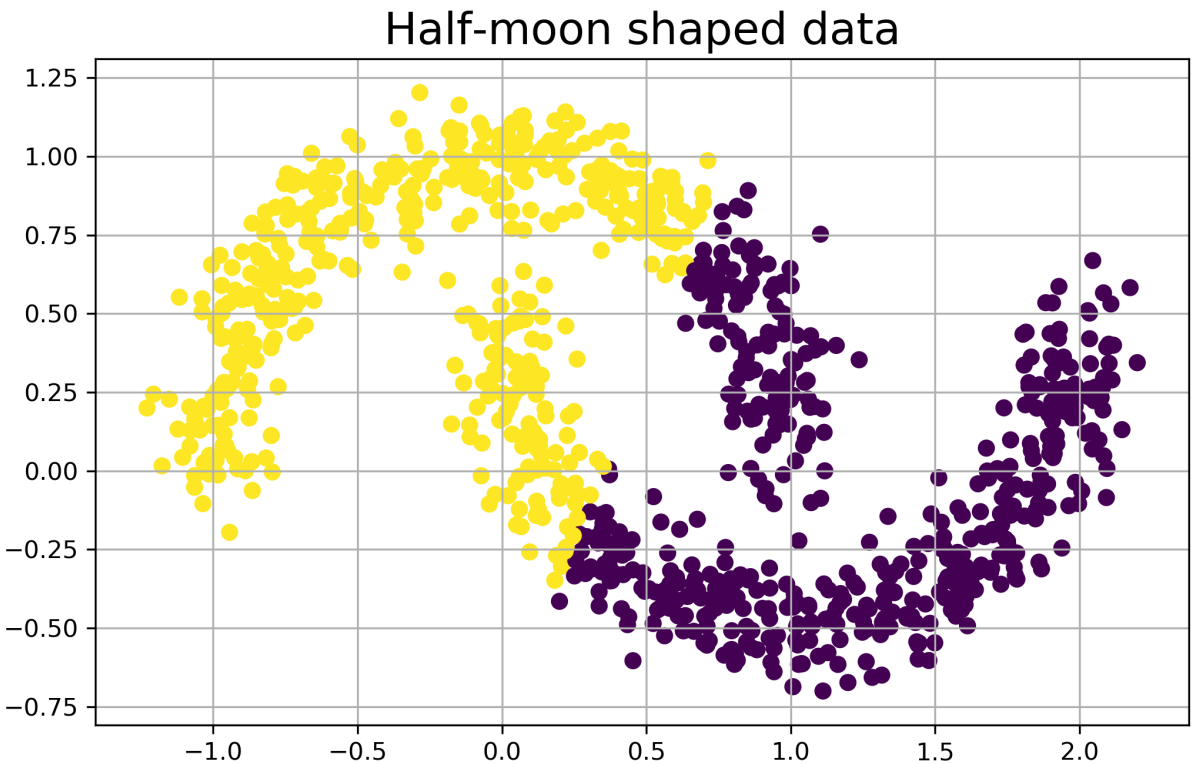


$$SSI_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

Let's run it for clustering moon data

```
# Fit K-Means Clustering on noise moon data
from sklearn import cluster
km=cluster.KMeans(n_clusters=2)
km.fit(noisy_moons)
km.labels_
# Plot moons
plt.figure(figsize=(8,5))
plt.title("Half-moon shaped data", fontsize=18)
plt.grid(True)
plt.scatter(noisy_moons[:,0],noisy_moons[:,1],c=km.labels_)
plt.show()
```

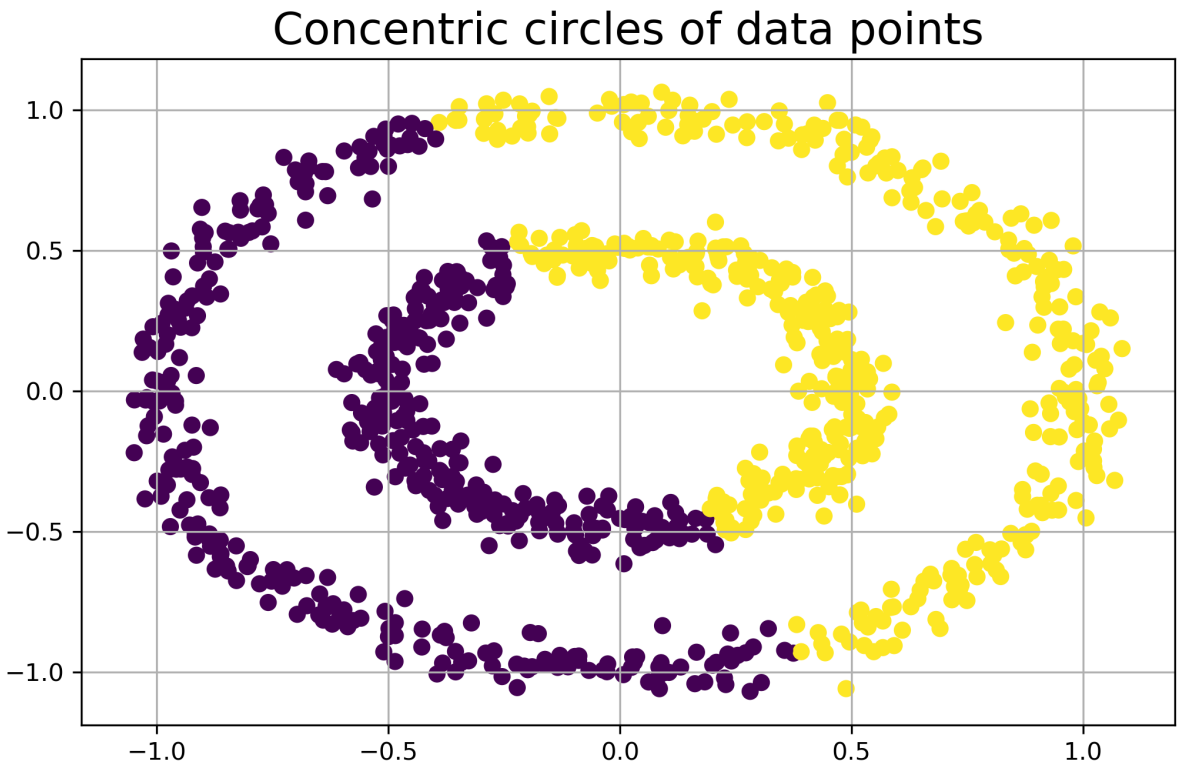
Two clusters on moon data



Let's run it for clsutering circle data

```
# Fit K-Means Clustering on noise Circle data
from sklearn import cluster
km=cluster.KMeans(n_clusters=2)
km.fit(noisy_circles)
km.labels_
# Plot circles
plt.figure(figsize=(8,5))
plt.title("Concentric circles of data points", fontsize=18)
plt.grid(True)
plt.scatter(noisy_circles[:,0],noisy_circles[:,1],c=km.labels_)
plt.scatter(noisy_moons[:,0],noisy_moons[:,1],c=km.labels_)
plt.show()
```

Two clusters on moon data



Disadvanatges of k-Means Clustering

- Difficult to predict k-value
- With global cluster, it didnt work well
- Different initial partitions can result in different final clusters

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

The DBSCAN algorithm should be used to find associations and structures in data that are hard to find manually but that can be relevant and useful to find patterns and predict trends.

Depends on two parameters

- eps: the minimum distance between two points. It means that if the distance between two points is lower or equal to this value (eps), these points are considered neighbors.
- minPoints: the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region.

DBSCAN: Advanatages and disadvanatges

Advanatages:

- Can discover arbitrarily shaped clusters
- Find cluster completely surrounded by different clusters

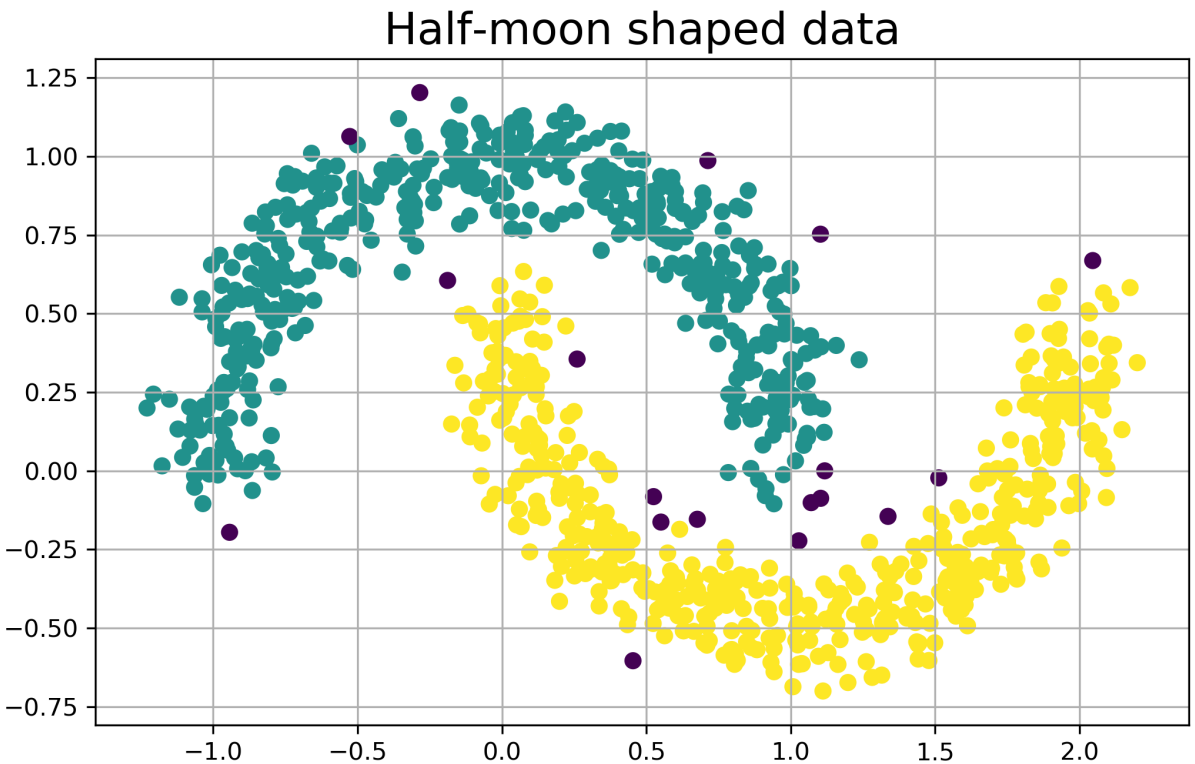
Disadvanatges

- Datasets with altering densities are tricky
- Sensitive on two parameters

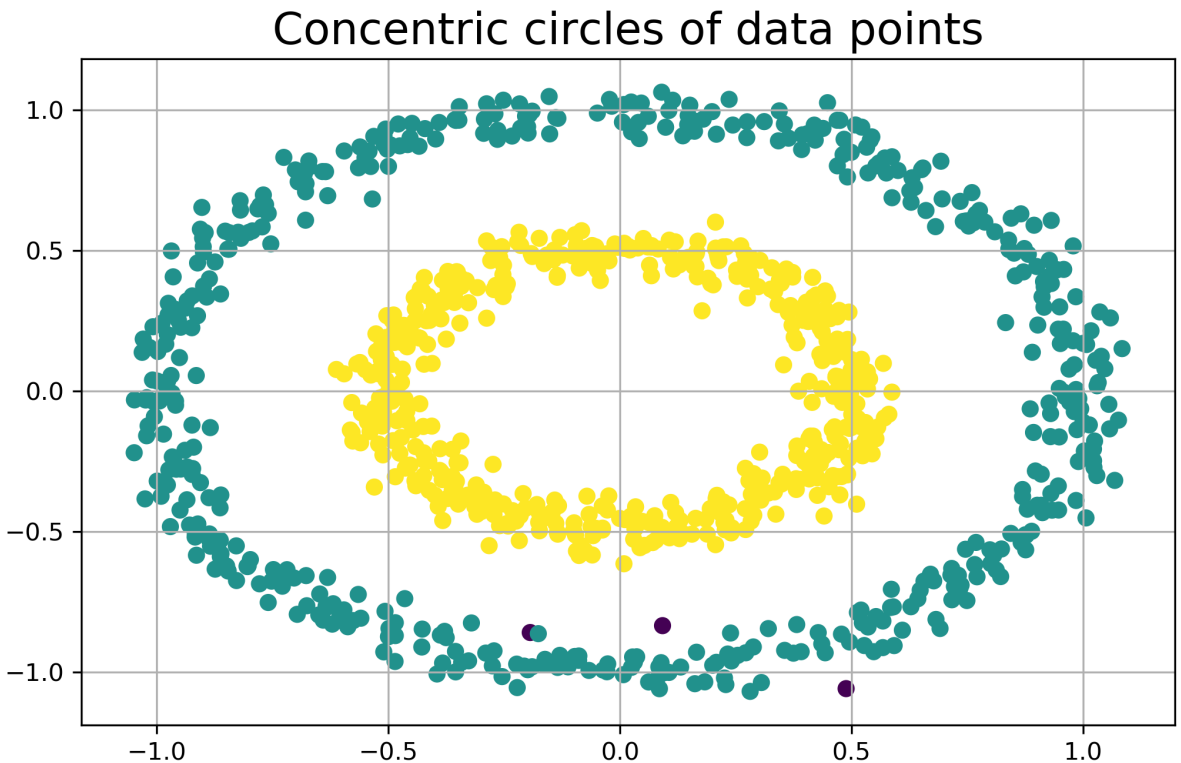
Let's run it for clustering moon data with DBSCAN

```
# Fit DBSCAN Clustering on moon data
from sklearn import cluster
# The maximum distance between two samples for them
#  to be considered as in the same neighborhood.
dbs = cluster.DBSCAN(eps=0.1)
dbs.fit(noisy_moons)
dbs.labels_
# Plot circles
plt.figure(figsize=(8,5))
plt.title("Half-moon shaped data", fontsize=18)
plt.grid(True)
plt.scatter(noisy_moons[:,0],noisy_moons[:,1],c=dbs.labels_)
plt.savefig('DBSCAN_MOON.png', dpi=300)
plt.show()
```

Two clusters on moon data using DBSCAN



Two clusters on circle data using DBSCAN



Principal component analysis (PCA)

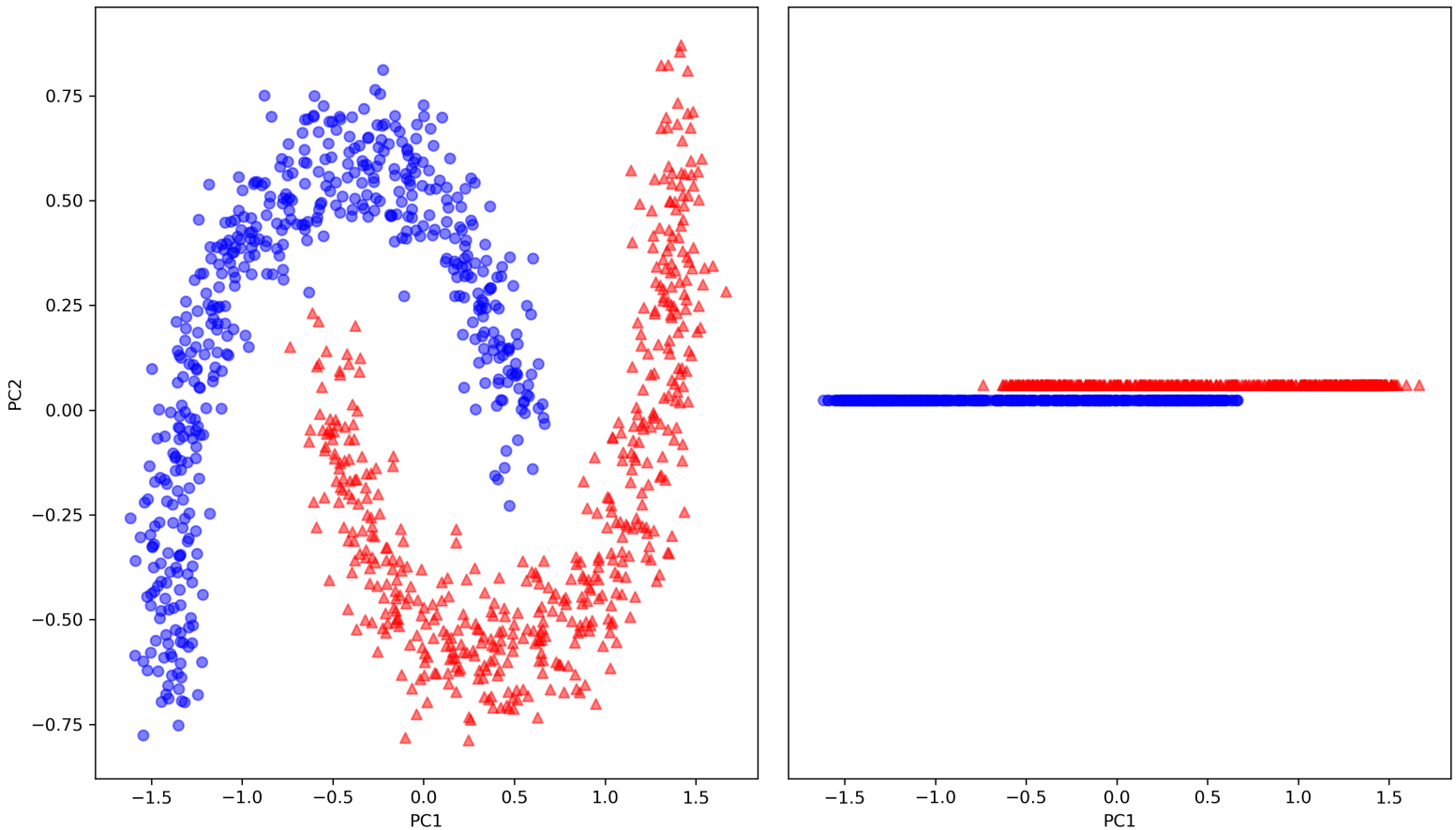
PCA is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize

- It is an orthogonal transformation to convert a set of correlated variables into a set of values of linearly uncorrelated variables called principal components, with the goal of finding the best summary of the data using a limited number of PCs.

Using PCA

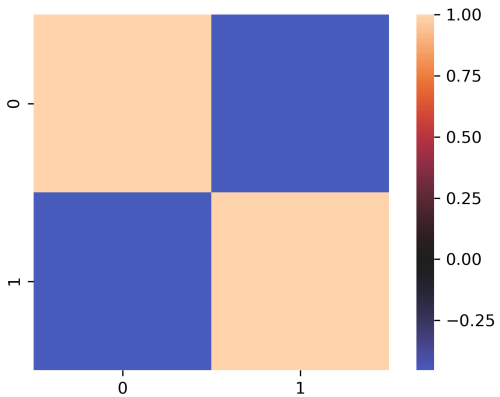
- PCA is not the only algorithm to do dimensionality reduction
- You can also use PCA for doing predictions
- Each PCA component is a linear combination of the input features
- So we can plot a correlation plot and see how they are related

Before and After PCA

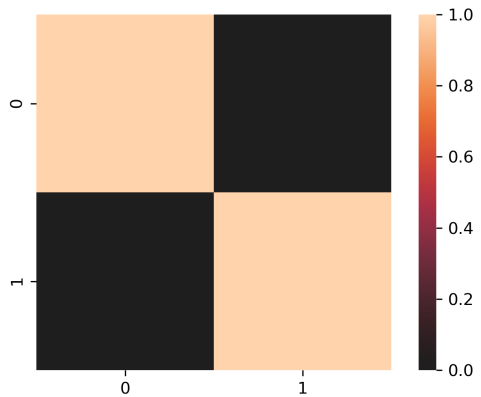


Correlation plot

Before PCA on Moon Data



After PCA on Moon Data



Kernel PCA

Is an extension of PCA using techniques of kernel methods. Using a kernel, the originally linear operations of PCA are performed in a reproducing kernel Hilbert space.

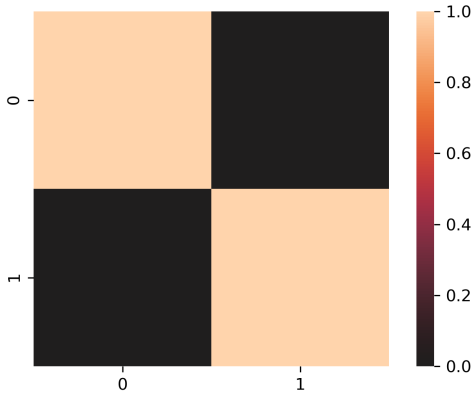
- Kernel PCA just performs PCA in a new space
- It uses Kernel trick to find principal components in different space (Possibly High Dimensional Space)
- Kernel PCA has been demonstrated to be useful for novelty detection and image de-noising
- PCA allow us to reconstruct pre-image and sometimes it may not be possible in KPCA

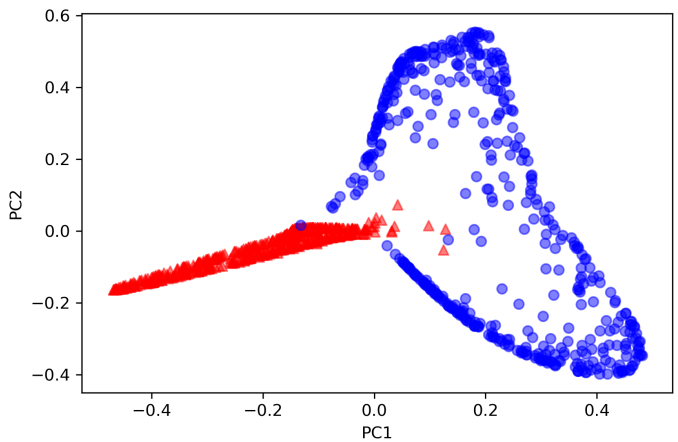
Let's run it for Kernel PCA on moon data

```
# Kernal PCA on moon data
from sklearn.decomposition import KernelPCA
scikit_kpca = KernelPCA(n_components=2, kernel='rbf', gamma=15)
noisy_moons_kpca = scikit_kpca.fit_transform(noisy_moons)

# Plot the moon data after KPCA trick
plt.scatter(noisy_moons_kpca[moon_labels == 0, 0], noisy_moons_kpca[moon_labels == 0, 1],
            color='red', marker='^', alpha=0.5)
plt.scatter(noisy_moons_kpca[moon_labels == 1, 0], noisy_moons_kpca[moon_labels == 1, 1],
            color='blue', marker='o', alpha=0.5)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.tight_layout()
plt.savefig('KPCA_MOON.png', dpi=300)
```

Kernal PCA plot

Kernal PCA corelation matrix heat map





Moon data after performing KPCA

Outlier detection

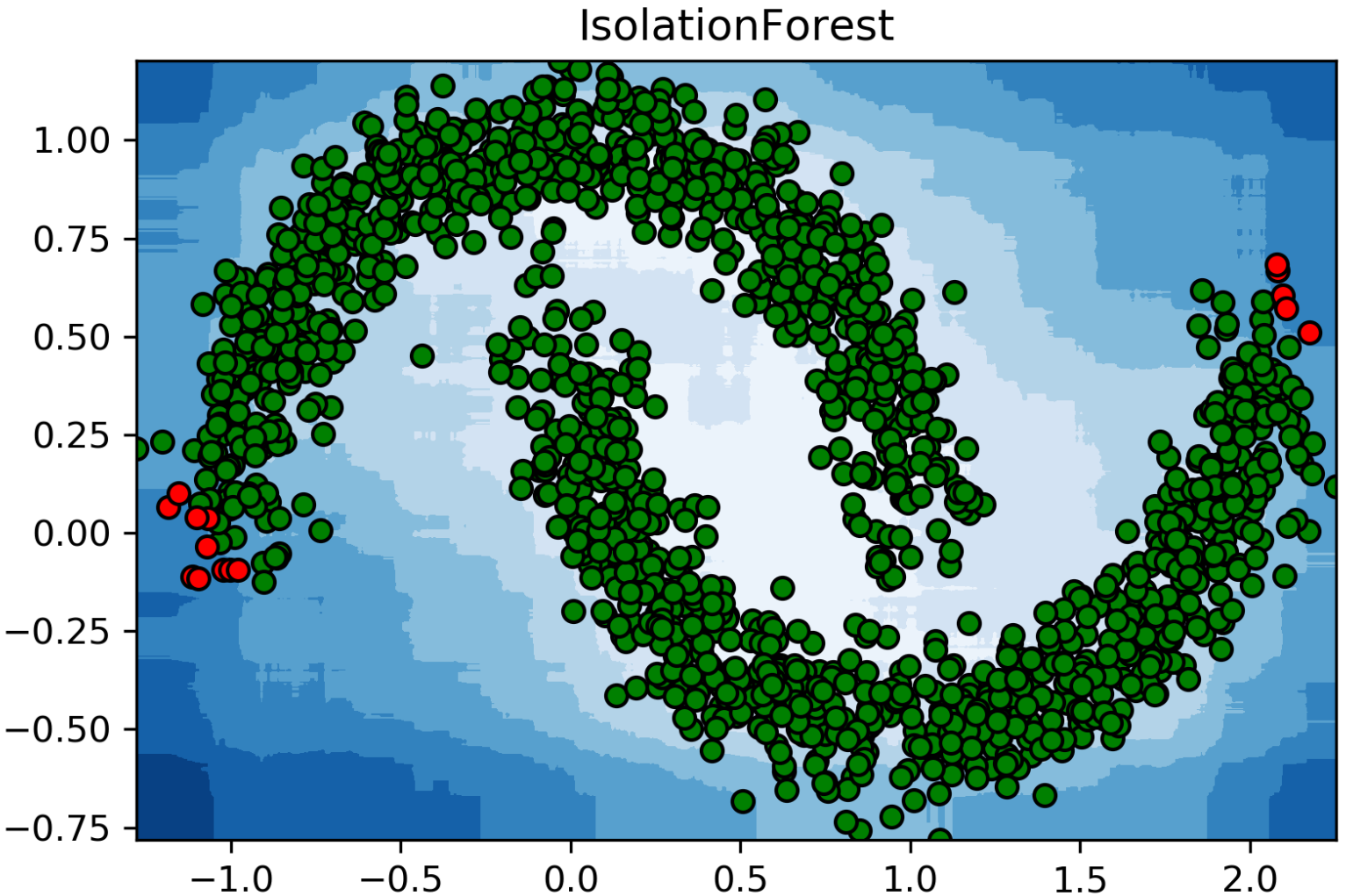
- You want to check which observations do not fit in with the rest
- Isolations forests
  - Keep splitting features until at random until every observation is in it's own tree leaf
  - Observations with short paths are treated

Isolation forests code

```
from sklearn.ensemble import IsolationForest

clf = IsolationForest(max_samples=100, contamination = 0.01)
clf.fit(noisy_moons)
y_pred_train = clf.predict(noisy_moons)

pos = y_pred_train > 0
neg = y_pred_train < 0

xx, yy = np.meshgrid(np.linspace(min((noisy_moons[:, 0])), max((noisy_moons[:, 0])), 500), np.linspace(min(
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.title("Isolation Forest")
plt.contourf(xx, yy, Z, cmap=plt.cm.Blues_r)

b1 = plt.scatter(noisy_moons[pos][:, 0], noisy_moons[pos][:, 1], c='green', edgecolor='k')
b2 = plt.scatter(noisy_moons[neg][:, 0], noisy_moons[neg][:, 1], c='red', edgecolor='k')

plt.axis('tight')

plt.xlim((xx.min(), xx.max()))
plt.ylim((yy.min(), yy.max()))

print(pos.sum())
print(neg.sum())
```

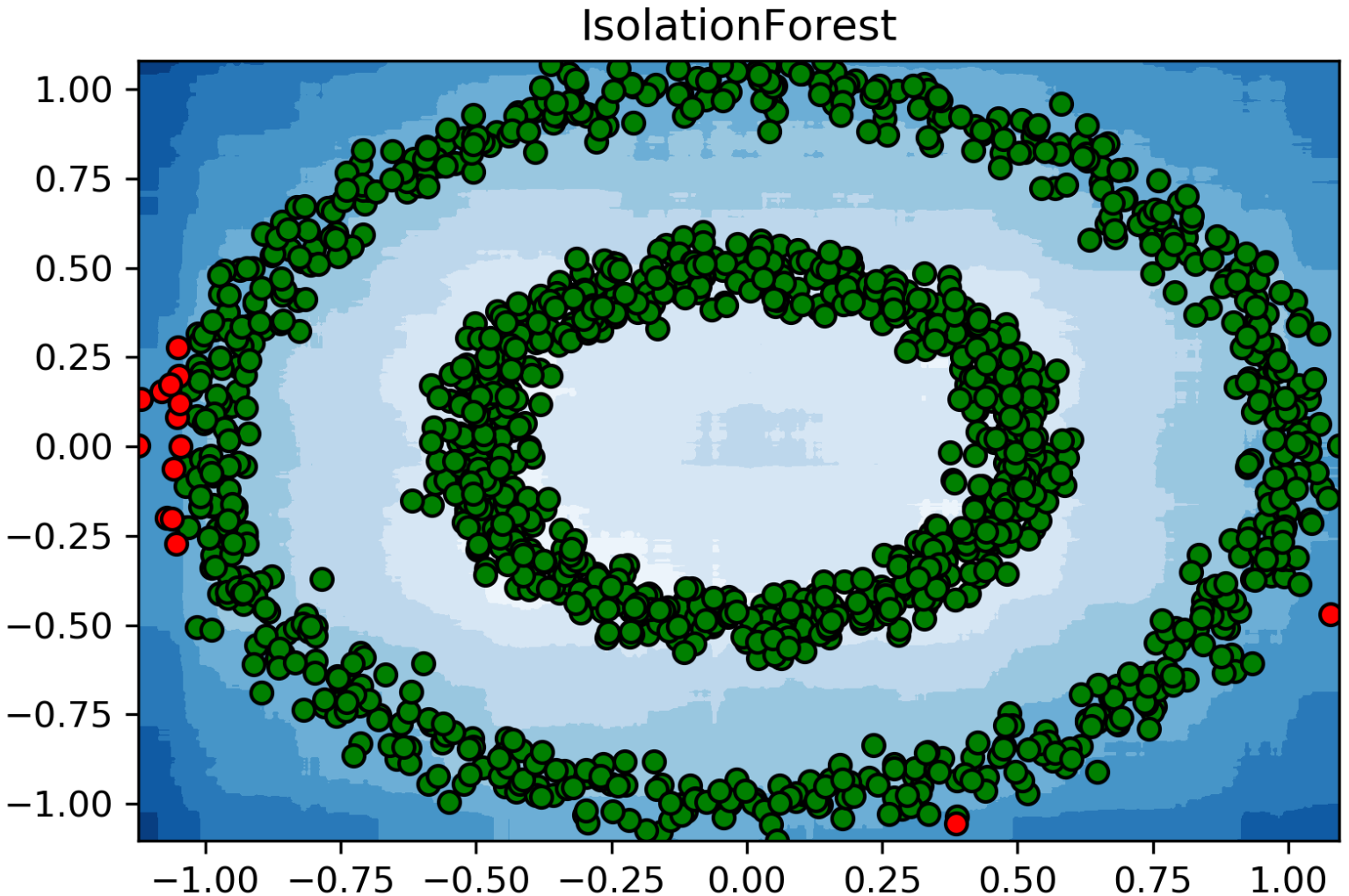
Outlier detection results on moon data

1485 Normal predictions, 15 Outliers



Outlier detection results on circle data

1485 Normal predictions, 15 Outliers



Let's dive into real world data

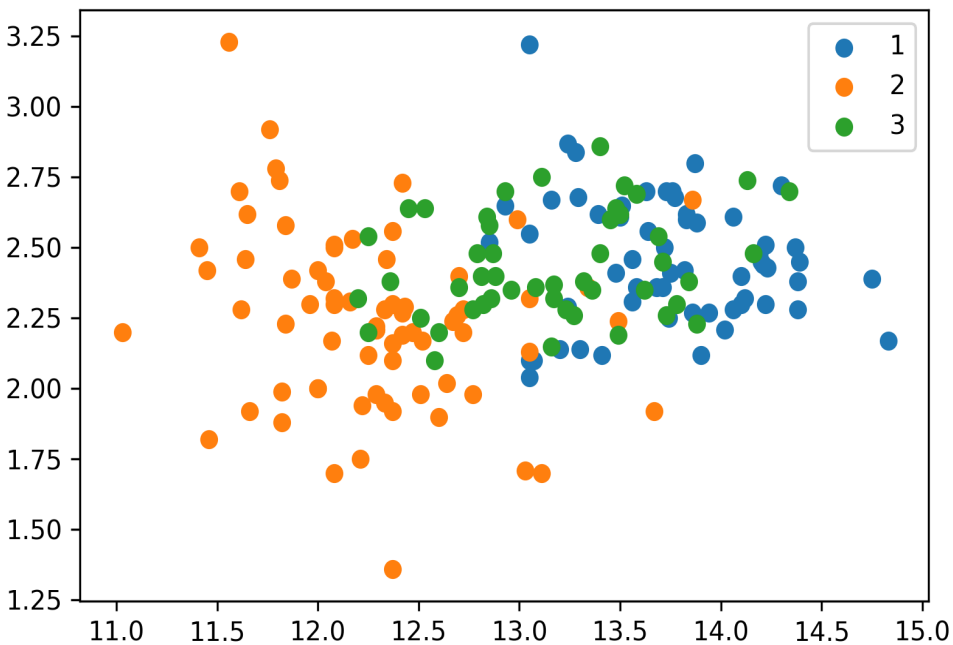
"This data set is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines"

`https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data}`

'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium','Total phenols', 'Flavanoids', 'Nonflavanoid phenols','Proanthocyanins', 'Color intensity', 'Hue','OD280/OD315 of diluted wines', 'Proline'

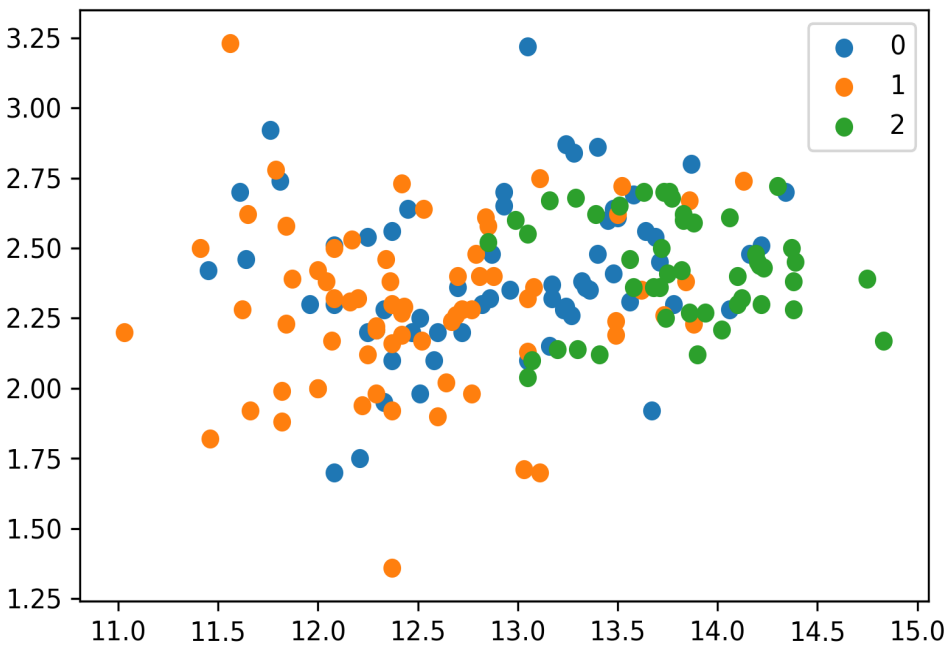
Let's cluster the wine dataset

- Clustering on true labels
- Number of clusters = 3 (three type of wines)



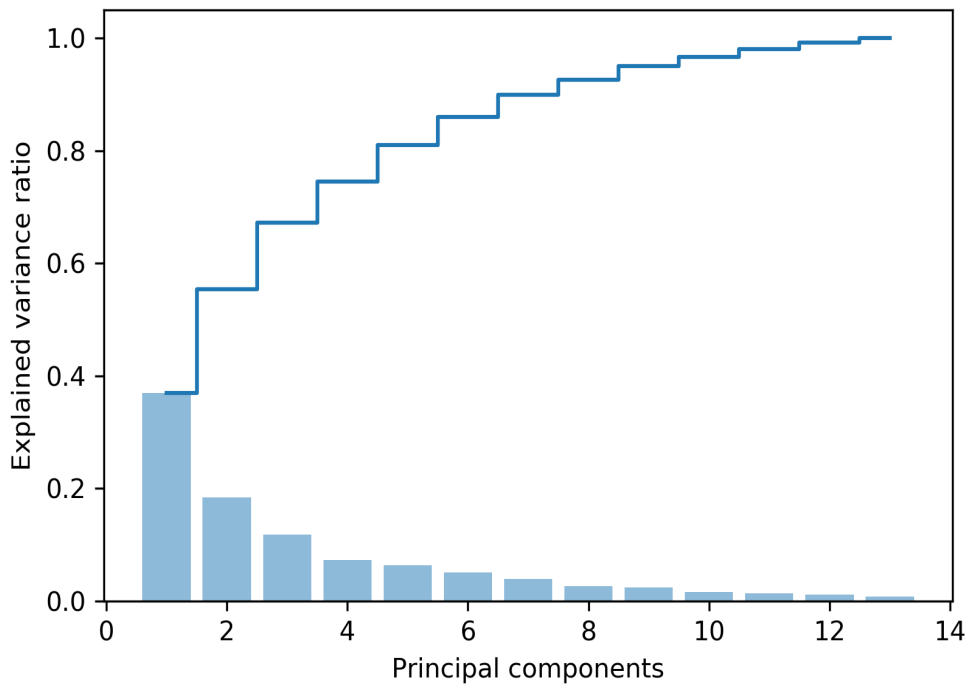
Let's cluster the wine dataset

- Clustering on unlabled data using two variables ['Alcohol'and 'Ash']
- Number of clusters = 3 (three type of wines)



Let's do PCA on wine data to reduce dimensionaity

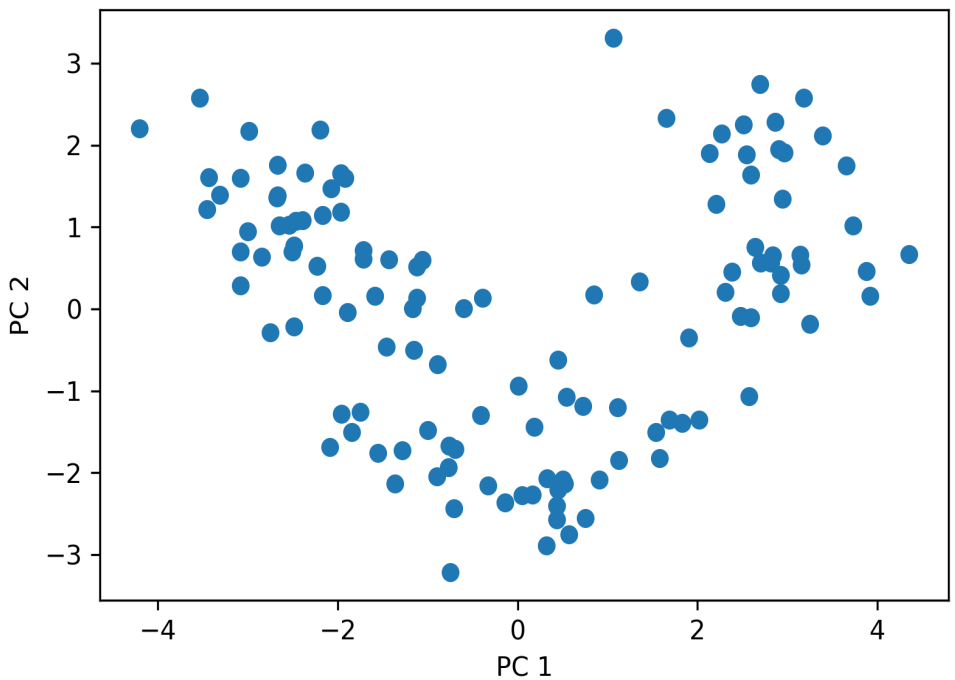
We will get Principal components on X-axis and explained variance ratio on Y-axis



Note: Here the data is splitted into 70:30 ratio

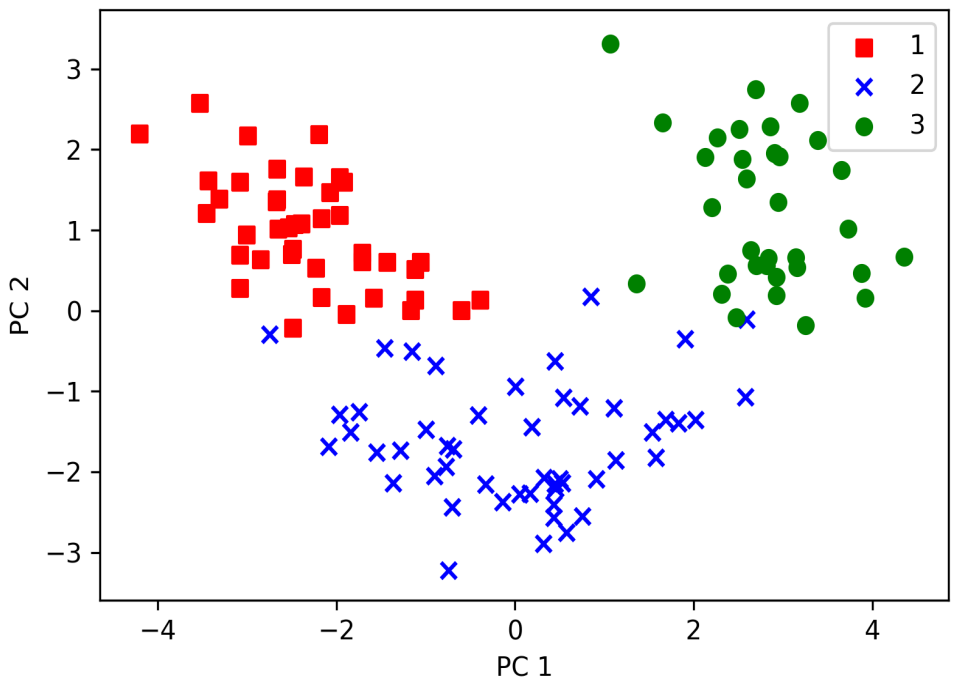
Let's do PCA ...

We will get plot on two best princial components (Unsupervised mode)



Let's do PCA ...

We will get plot on two best princial components (Supervised mode)

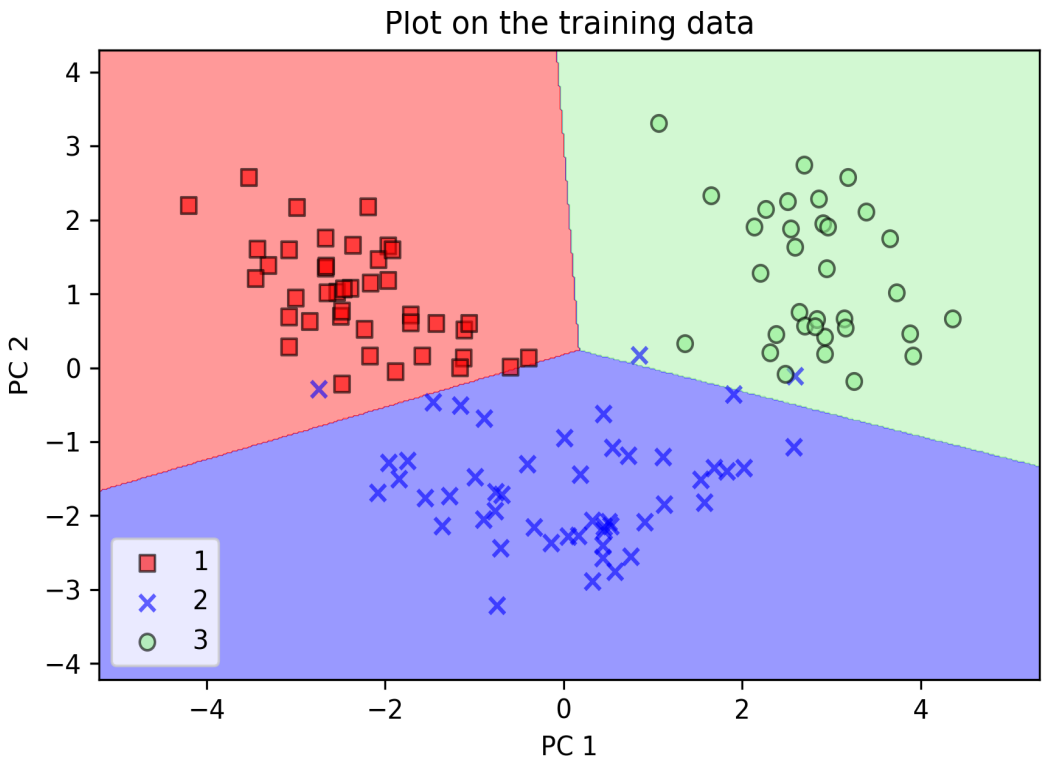


Let's train a Logistic Regression Model to classify the test data

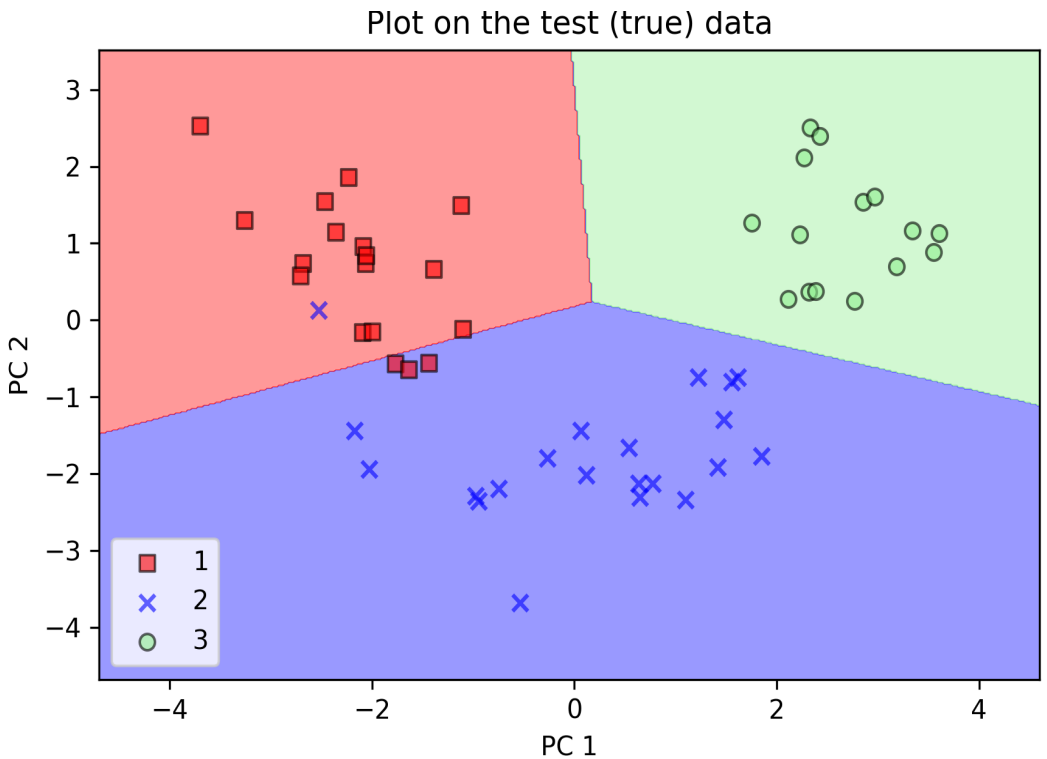
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_std)
X_test_pca = pca.transform(X_test_std)

lr = LogisticRegression()
lr = lr.fit(X_train_pca, y_train) # Train a Logistic regression model
pred_lab=lr.predict(X_test_pca) # Test on the test data
print("Accuracy on test is = %f" % (100 * accuracy_score(y_test, pred_lab)))
```

Accuracy on test data is = 92.592593

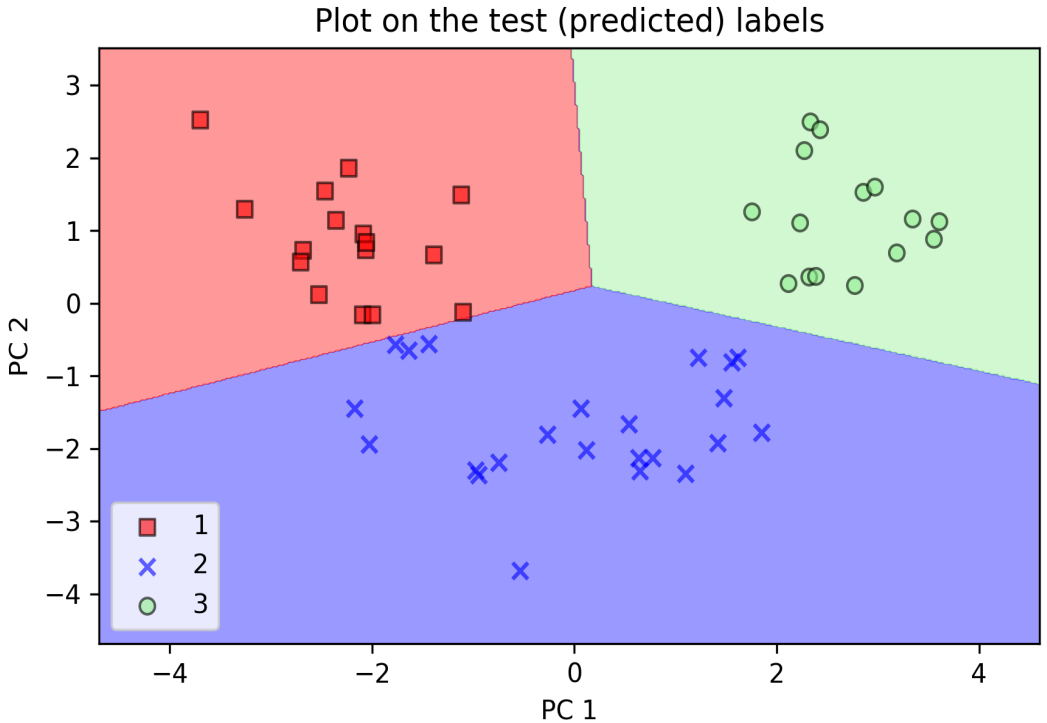
Plot on the training data ...



Plot on the test (true) data ...



Plot on the test (predicted) labels ...



Conclusion

- We have seen a set of methods for understanding the data without an outcome signal to guide us
- There is a revolution currently going on in this field, we will cover it after neural networks
- Some of the methods useful even if you have a signal, e.g. do PCA/KMeans on the data and then feed them into a classifier