

Regression Models

Data Science & Business Analytics

Lesson 4

Lesson Plan

- Explain what is regression
- Present 6 most common algorithms
- How to transform a regression problem into a classification model
- Example
- Explain is unsupervised Learning
- Explain most common algorithms
- Example



Linear Regression

In regression model the idea is to predict continuous data. For example the age, or salary of someone.

The formula is simple linear regression is $y = m x + c$.

Like other models, you can analyse how good it is. In this example this is done by calculating the error (next slide)

In python

```
from sklearn.linear_model import  
LinearRegression
```

```
regressor = LinearRegression()
```

```
regressor.fit(X_train, y_train)
```



Simple Linear Regression

Regression error

The most common is the MSE (Mean Square error). Represented in the picture in the right.

Other errors are:

1. Root Mean Squared Error

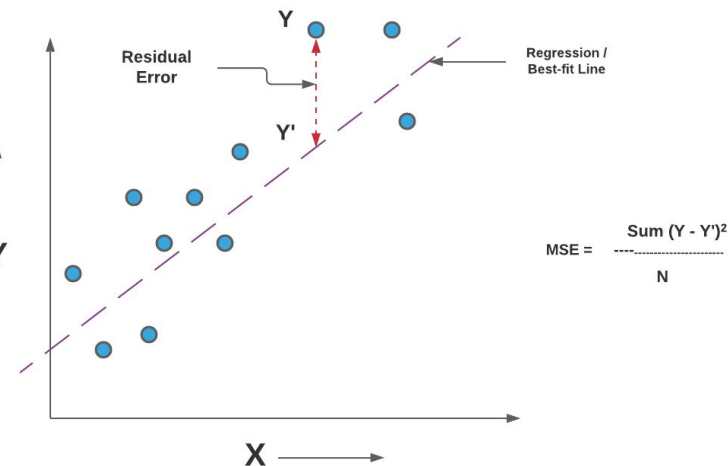
$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

```
from sklearn.metrics import mean_squared_error
```

2. Mean Absolute Error

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

```
from sklearn.metrics import mean_absolute_error
```



Multiple Linear Regression

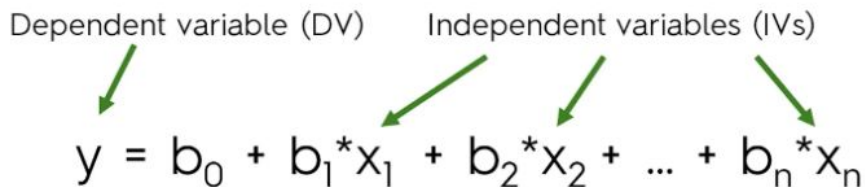
This is very similar to the one dependent variable linear regression, but generalized to multiple variables.

```
from sklearn.linear_model import LinearRegression
```

```
regressor = LinearRegression()
```

```
regressor.fit(X_train, y_train)
```

Dependent variable (DV) Independent variables (IVs)



The diagram shows the equation $y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n * x_n$. Above the equation, the text "Dependent variable (DV)" has a green arrow pointing to the variable y . The text "Independent variables (IVs)" has three green arrows pointing to the variables x_1 , x_2 , and x_n .

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n * x_n$$

Multiple Linear Regression Equation

Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly_reg = PolynomialFeatures(degree = 3)
```

```
X_poly = poly_reg.fit_transform(X)
```

```
lin_reg = LinearRegression()
```

```
lin_reg.fit(X_poly, y)
```

$$y = b_0 + b_1x_1 + b_2x_1^2 + \dots + b_nx_1^n$$

Polynomial Regression Equation

Lasso Regression

The purpose of lasso and ridge is to stabilize the vanilla linear regression and make it more robust against outliers, overfitting, and more.

Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients.

$$\sum_{i=1}^n (y_i - \sum_j x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

- When $\lambda = 0$, no parameters are eliminated. The estimate is equal to the one found with linear regression.
- As λ increases, more and more coefficients are set to zero and eliminated (theoretically, when $\lambda = \infty$, all coefficients are eliminated).
- As λ increases, bias increases.
- As λ decreases, variance increases.

[https://www.statisticshowto.com/lasso-regression/#:~:text=Lasso%20regression%20is%20a%20type,i.e.%20models%20with%20fewer%20parameters\).](https://www.statisticshowto.com/lasso-regression/#:~:text=Lasso%20regression%20is%20a%20type,i.e.%20models%20with%20fewer%20parameters).)

Ridge Regression

Least squares regression isn't defined at all when the number of predictors exceeds the number of observations; It doesn't differentiate "important" from "less-important" predictors in a model, so it includes all of them. This leads to overfitting a model and failure to find unique solutions. Least squares also has issues dealing with multicollinearity in data.

Ridge regression belongs a class of regression tools that use L2 regularization.

L2 regularization adds an L2 penalty, which equals the square of the magnitude of coefficients. All coefficients are shrunk by the same factor (so none are eliminated). Unlike L1 regularization, L2 will not result in sparse models.

$$\text{Min}(\|Y - X(\theta)\|^2 + \lambda\|\theta\|^2)$$

- The bias increases as λ increases.
- The variance decreases as λ increases.

[https://www.statisticshowto.com/lasso-regression/#:~:text=Lasso%20regression%20is%20a%20type,i.e.%20models%20with%20fewer%20parameters\).](https://www.statisticshowto.com/lasso-regression/#:~:text=Lasso%20regression%20is%20a%20type,i.e.%20models%20with%20fewer%20parameters).)

Elastic Net

Elastic Net implements both Lasso and Ridge Regression so both

L1 and L2 can be choose independently.

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}} (\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1).$$

<https://www.statisticshowto.com/lasso-regression/#:~:text=Lasso%20regression%20is%20a%20type,i.e.%20models%20with%20fewer%20parameters>).

Support Vector Regression

In the Support Vector Regression Model, we have an ϵ -tube of width ϵ and a regression line in the middle of this tube. Our goal is to draw a line such that the sum of squared errors is minimized.

The errors are calculated by measuring the distance of points lying outside the tube from the closest point on the tube. The name Support Vector Regression comes from the fact that all points can be represented as vectors starting from the origin.

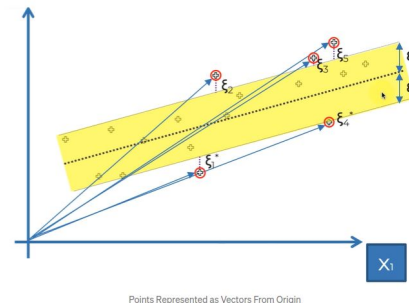
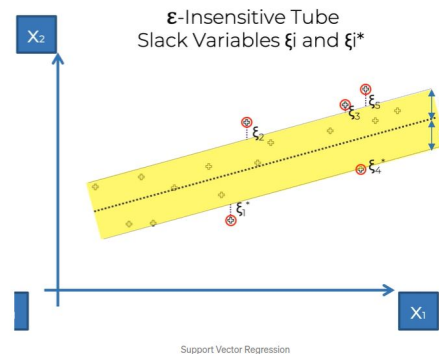
https://www.youtube.com/watch?v=-EjQWqHMsog&ab_channel=Iknowpython

https://www.youtube.com/watch?v=efR1C6CvhmE&ab_channel=StatQuestwithJoshStarmer

```
from sklearn.svm import SVR
```

```
regressor = SVR(kernel = 'rbf')
```

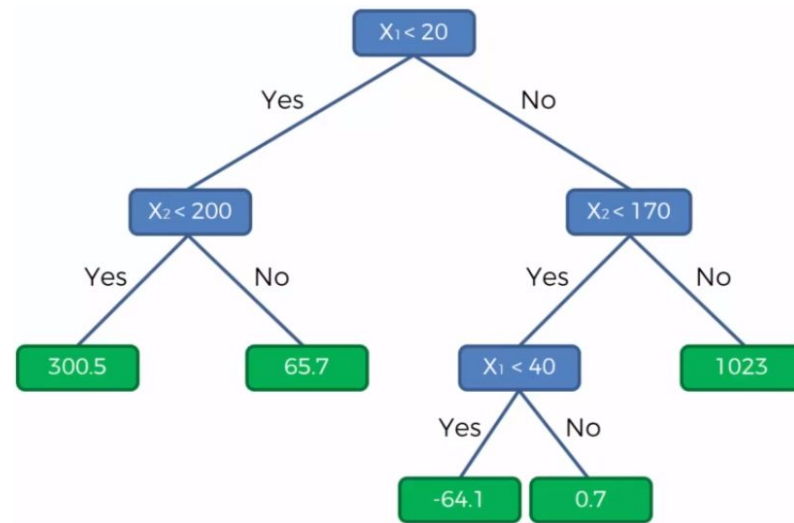
```
regressor.fit(X, y)
```



Decision Tree Regression

This same of Decision tree classification. But in this use case the output is the continuous variable.

```
from sklearn.tree import  
DecisionTreeRegressor  
  
regressor =  
DecisionTreeRegressor(random_state = 0)  
  
regressor.fit(X, y)
```

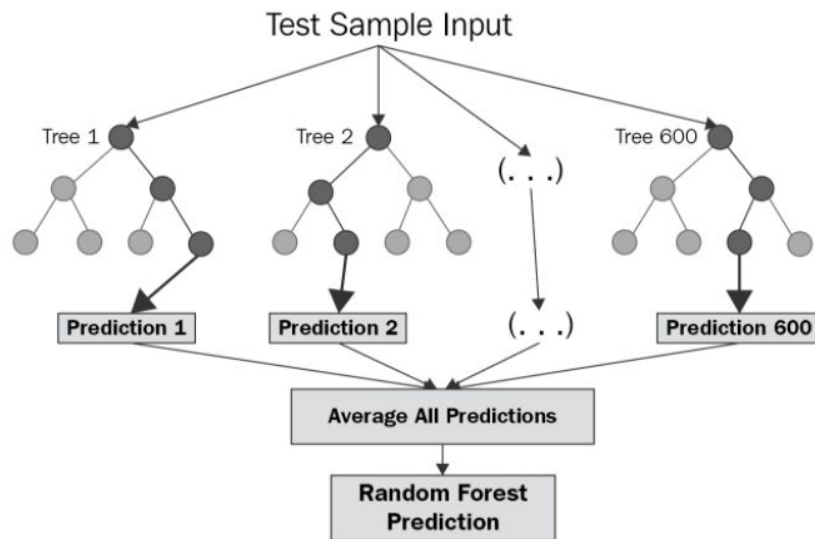


Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators =
10, random_state= 0)

regressor.fit(X, y)
```



Let's practice

1. Load Boston housing prices dataset
2. EDA
3. Check for outliers in the data
4. Remove them if needed
5. Let's compute correlation between features and predict label
6. Apply MinMax scaler to the dataset
7. Plot each feature against the predict label and plot a linear regression just for viz purposes
8. Check data skewness, and remove if bigger than 0.3
9. Let's apply the following models: linear regression, Lasso Regression, Ridge Regression, ElasticNet, DecisionTreeRegressor and SVR
10. Apply a 10 fold validation.

Unsupervised learning

In this kinds of models, the predict label is not provided, so the model works by trying to group the features considering the similarity. The goal is still to understand the underlying data distribution.

Unsupervised can be divided into:

- Clustering: Divide data into groups . For example, group cars by type
 - Algo: K-means
- Association: The idea here is to find the rules that describe large portions of data.
 - Apriori algorithm association

Not scope of the course

A third use case is to use unsupervised algorithms to reduce the number of dimensions in the data.



Supervised Learning

Unsupervised Learning

Dimensionality reduction

This technique is sometimes necessary when the number of features i.e. number of columns is very large. This creates very sparse matrices that make training very hard and slow. Also, it is very difficult to find a good solution.

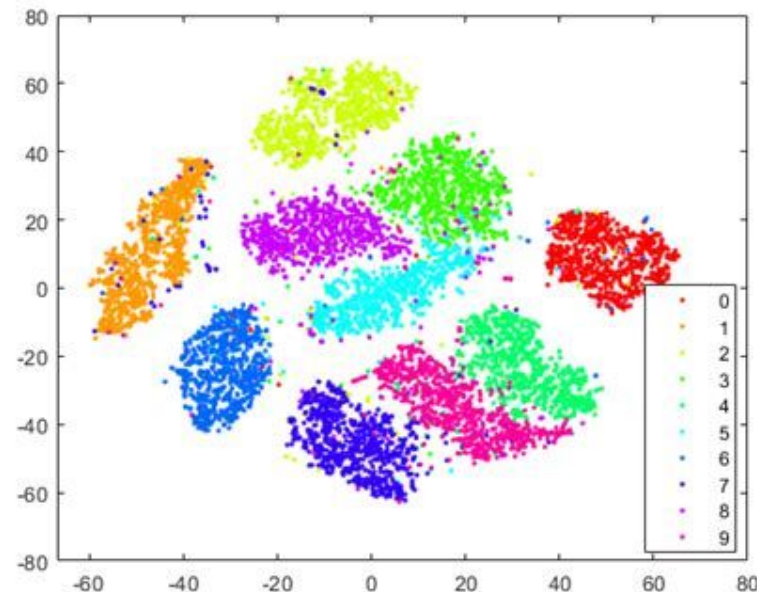
This is called “Dimensionality curse”

How the right, you can see what this looks like when we apply a tsne technique.

Another popular technique is PCA (Principal component analysis)

But there are many many more

<https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b>



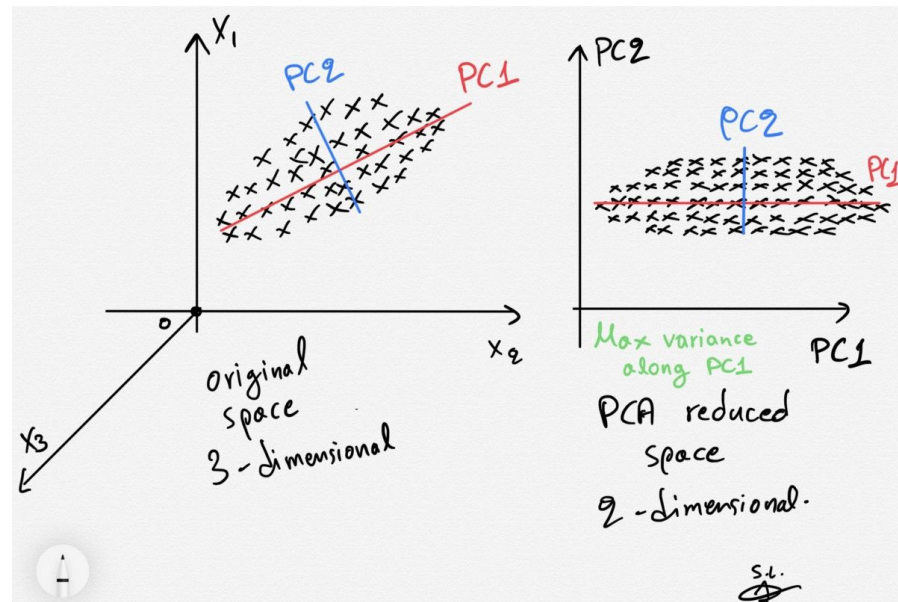
PCA

In this algorithm we search for the direction that have the highest variance. And after that, we project the data, in the subspace of highest variance.

To find those direction we use the eigenvectors, that we say in the first lecture. And the magnitude is compute using the eigenvalues.

5 MIN EXPLANANTION

https://www.youtube.com/watch?v=HMOI_lkzW08&ab_channel=S_tatQuestwithJoshStarmer

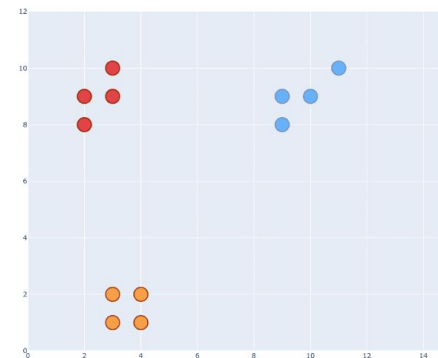
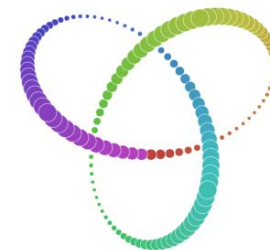
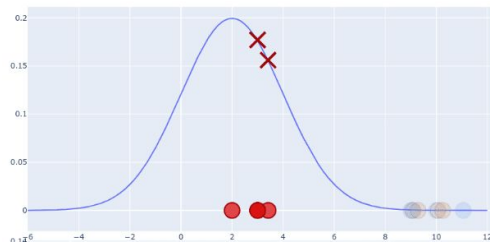


T-SNE

Like PCA the idea is to reduce the number of features. It was developed by Laurens van der Maaten and Geoffrey Hinton in 2008.

T-SNE is important because it works in data that is not separable by linear methods like PCA.

The base idea is to define the probability definition that mimics each group.



More info here

<https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a>

PCA (Exercise)

- Load iris dataset from url =

```
"https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

- Apply normalization into the data
- Extract two components using

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=2)
```

- Plot each component and the respectful labels
- Check explained variance using

```
pca.explained_variance_ratio_
```

- How to choose number of PC?
 - a. Choose by selection the sum of explained variance. Ideally should be higher than 90%

Clustering

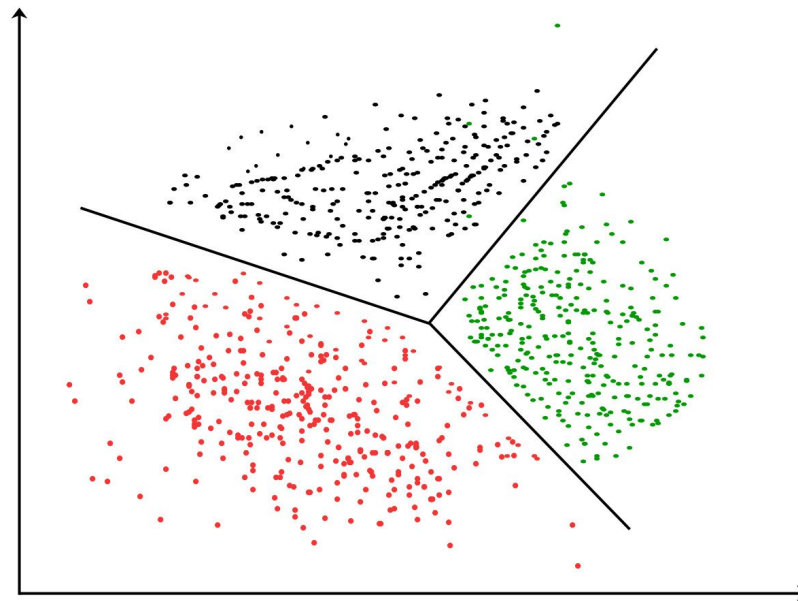
Clustering methods consist of the separation of the data by the similarity of the data.

There are plenty of methods, in this course we will cover:

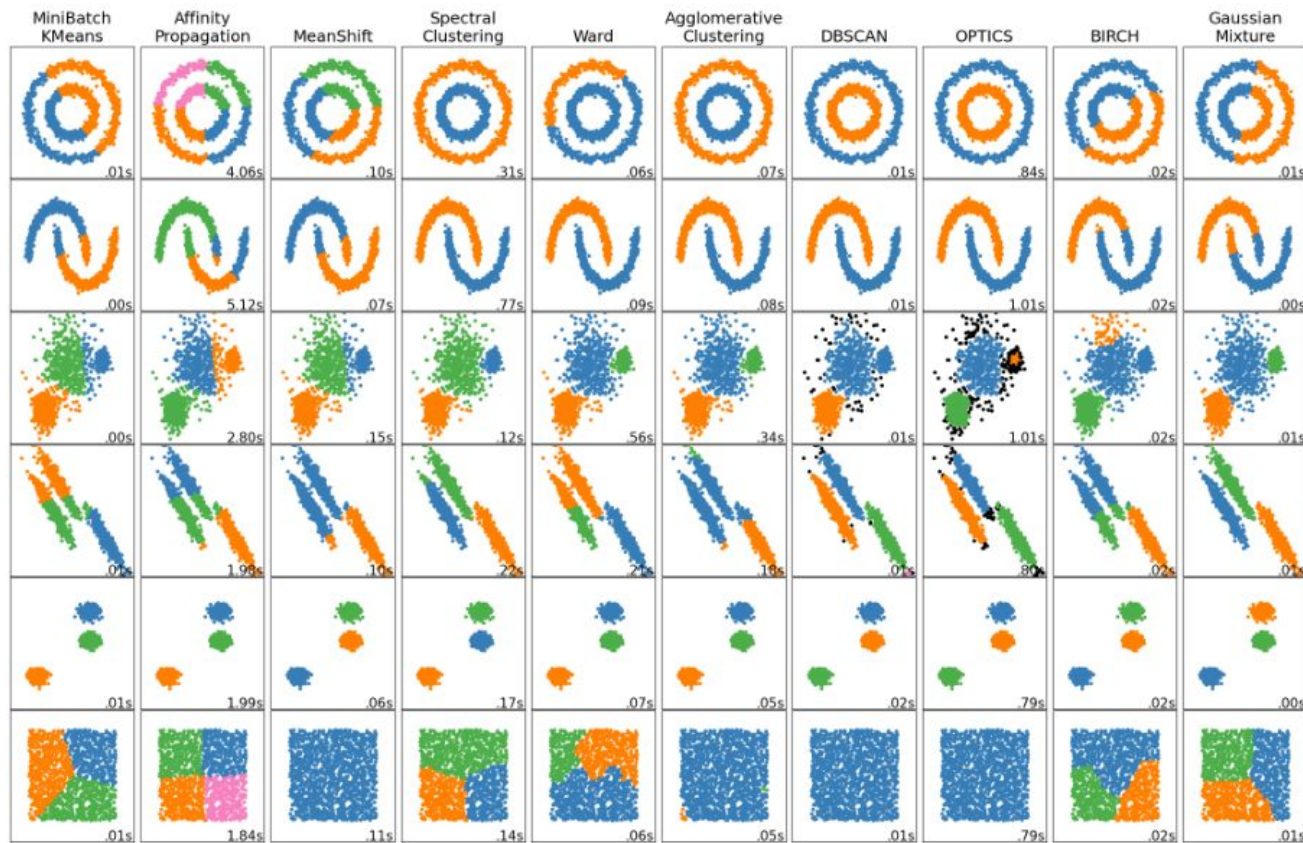
- K-means

There may exist use cases where we even do not know what to expect from the clusters.

There was one use case I did in Stratio, where the states meant machine working behavior patterns.



Clustering

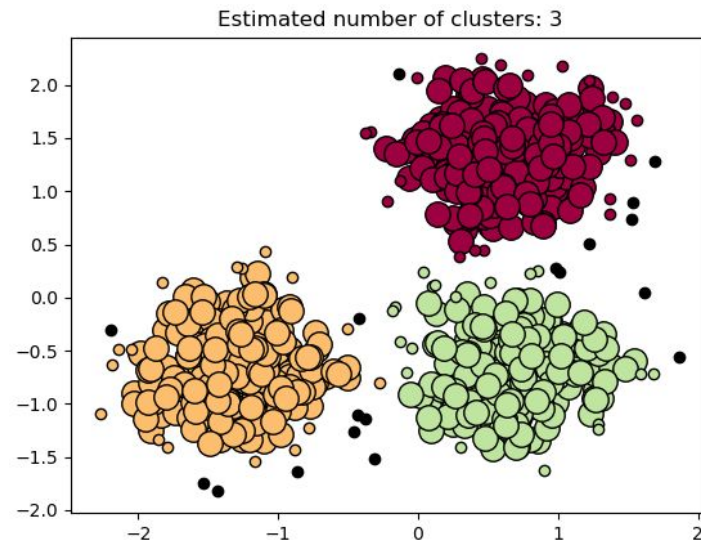


Density-Based Clustering

In this method, the clusters are created based upon the density of the data points which are represented in the data space. The regions that become dense due to the huge number of data points residing in that region are considered as clusters.

The data points in the sparse region (the region where the data points are very less) are considered as noise or outliers. The clusters created in these methods can be of arbitrary shape.

Example: [DBSCAN](#)



Models

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large <code>n_samples</code> , medium <code>n_clusters</code> with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with <code>n_samples</code>	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with <code>n_samples</code>	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium <code>n_samples</code> , small <code>n_clusters</code>	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large <code>n_samples</code> and <code>n_clusters</code>	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large <code>n_samples</code> and <code>n_clusters</code>	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large <code>n_samples</code> , medium <code>n_clusters</code>	Non-flat geometry, uneven cluster sizes, outlier removal, transductive	Distances between nearest points
OPTICS	minimum cluster membership	Very large <code>n_samples</code> , large <code>n_clusters</code>	Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large <code>n_clusters</code> and <code>n_samples</code>	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points

Partition methods - K-means

The K-Means algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares (see below).

This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

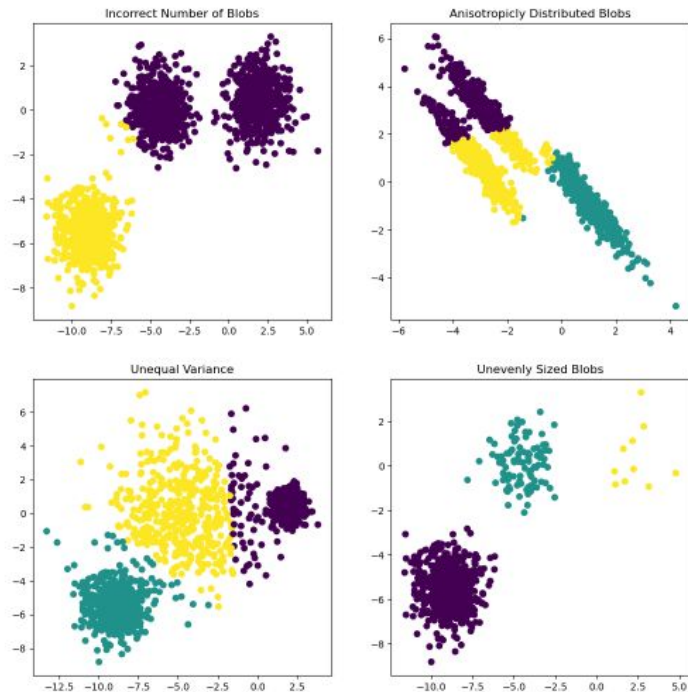
The k-means algorithm divides a set of N samples X into K disjoint clusters, each described by the mean of the samples in the cluster. The means are commonly called the cluster “centroids”; note that they are not, in general, points from X , although they live in the same space.

The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion:

$$\sum_{i=0}^n \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

K-means drawbacks

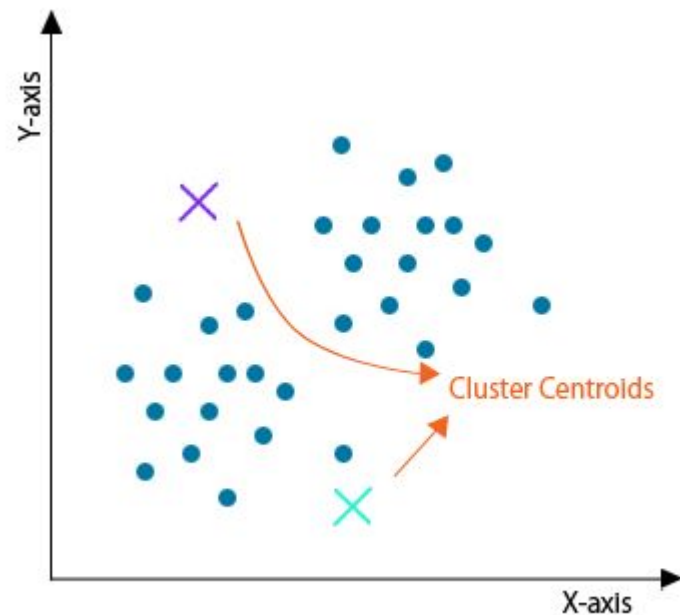
- Inertia makes the assumption that clusters are convex and isotropic, which is not always the case. It responds poorly to elongated clusters, or manifolds with irregular shapes.
- Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called “curse of dimensionality”). Running a dimensionality reduction algorithm such as Principal component analysis (PCA) prior to k-means clustering can alleviate this problem and speed up the computations.



K-Means - Step 1

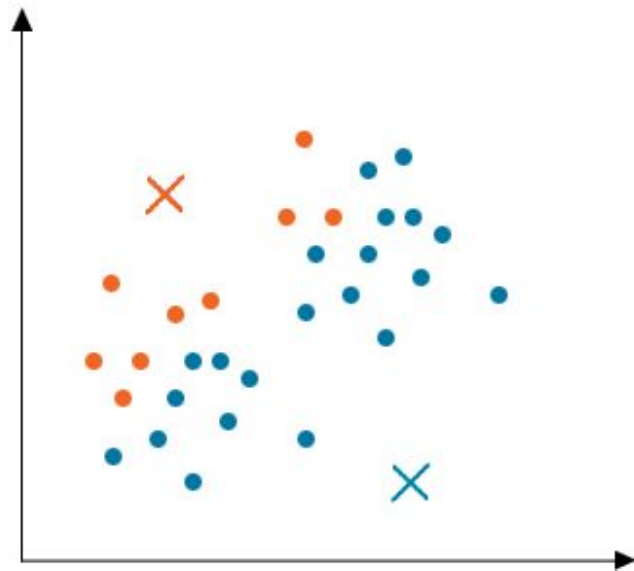
When we are starting off the algorithm, we randomly assign the number of desired centroids in the data space. Bear in mind that the location of the centroids can be different, depending on the method utilized.

The only constraint to have in mind is the number of data points. The number of clusters cannot be bigger.



K-Means - Step 2

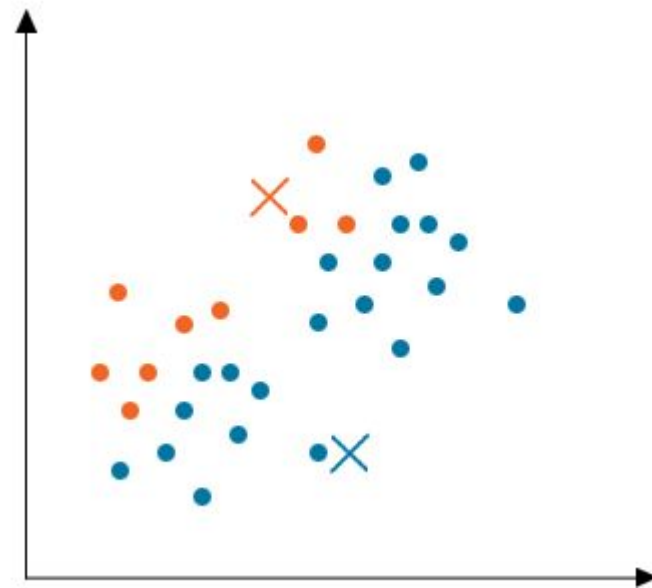
In this step, data is divided based on the distance between each point to the centroid. The goal is to minimize that distance.



K-Means - Step 3

Once we divide the data, we update the location of the centroid.

Normally a average is computed.



Why K-means?

Below are the advantages mentioned:

- It is fast
- Robust
- Easy to understand
- Comparatively efficient
- If data sets are distinct, then gives the best results
- Produce tighter clusters
- When centroids are recomputed, the cluster changes.
- Flexible
- Easy to interpret
- Better computational cost
- Enhances Accuracy
- Works better with spherical clusters

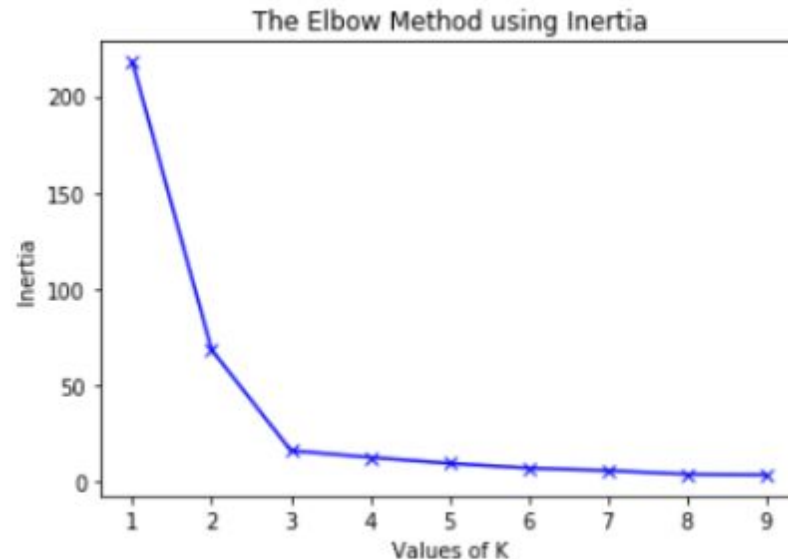
Below are the disadvantages mentioned:

- Needs prior specification for the number of cluster centers
- If there are two highly overlapping data, then it cannot be distinguished and cannot tell that there are two clusters
- With the different representations of the data, the results achieved are also different
- Euclidean distance can unequally weigh the factors
- It gives the local optima of the squared error function
- Sometimes choosing the centroids randomly cannot give fruitful results
- It can be used only if the meaning is defined
- Cannot handle outliers and noisy data
- Do not work for the non-linear data set
- Lacks consistency
- Sensitive to scale
- If very large data sets are encountered, then the computer may crash.
- Prediction issues

How to choose a clustering model

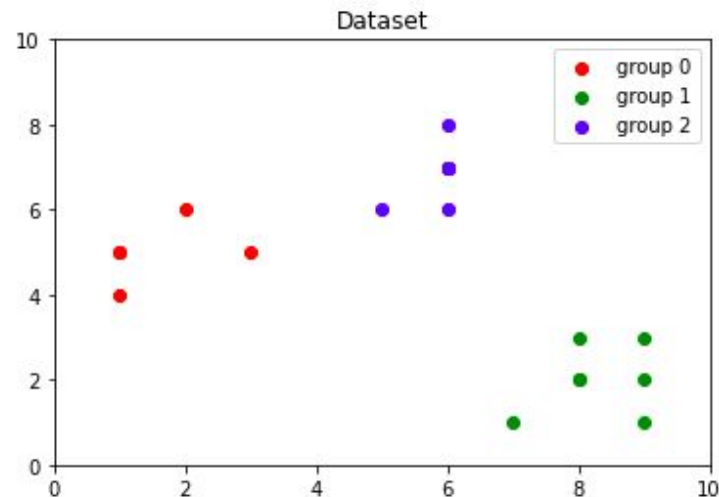
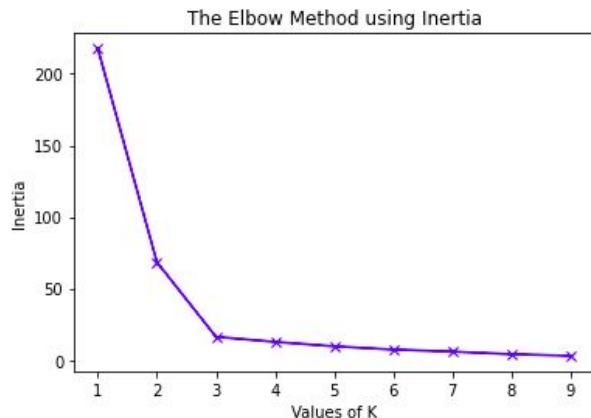
The technique is called elbow method. Because it looks like an elbow :=

The idea is to iterate over several different values of k , and check when the increase in cluster no longer helps.



Example of application

<https://colab.research.google.com/drive/1EE0THCPBFYxNFnLN8a6QR5DTcXA1GB3i?usp=sharing>



Clustering exercise

1. Load car dataset
2. EDA (show feature histogram)
3. Fill missing values if needed
4. Plot T-SNE
5. Plot elbow chart
6. Choose n of cluster
7. Plot with different clusters and compare
8. Plot results