Assignment No. 1

Title

Assignment based on Prediction

Problem Definition:
Predict the price of the Uber ride from a given pickup point to the agreed drop-off location.
Perform following tasks:
1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.
4. Implement linear regression and random forest regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc.
.

Prerequisite:

Basic of Python, Data Mining Algorithm, Concept of Prediction, Regression

Software Requirements:

Anaconda with Python 3.7

Theory Concepts:

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1: intercept

θ_2: coefficient of x

Random forest algorithm

Random forest algorithm builds a forest in the form of an ensemble of decision trees which adds more randomness while growing the trees. While splitting a node, the algorithm searches for the best features from the random subset of features which adds more diversity, thereby resulting in a better model.

Regression models are used to quantify the relationship between one or more predictor variables and a response variable.

Whenever we fit a regression model, we want to understand how well the model "fits" the data. In other words, how well is the model able to use the values of the predictor variables to predict the value of the response variable?

Two metrics that statisticians often use to quantify how well a model fits a dataset are the root mean squared error (RMSE) and the R-squared (R^2), which are calculated as follows:

RMSE: A metric that tells us how far apart the predicted values are from the observed values in a dataset, on average. The lower the RMSE, the better a model fits a dataset.

It is calculated as:

$$RMSE = \sqrt{\Sigma(P_i - O_i)^2 / n}$$

where:

- Σ is a symbol that means "sum"
- P_i is the predicted value for the i^{th} observation
- O_i is the observed value for the i^{th} observation
- n is the sample size

R^2: A metric that tells us the proportion of the variance in the response variable of a regression model that can be explained by the predictor variables. This value ranges from 0 to 1. The higher the R^2 value, the better a model fits a dataset

Conclusion

In this way we predicted and Performed following tasks:
1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.

4. Implement linear regression and random forest regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc.and analysed Classified the email using the binary classification method.

Assignment Questions

1. Why is random forest better than logistic ?
2. How does random forest regression algorithms works?
3. What are the linear regression and random forest?
4. What is R2, RMSE?
5. What is regression?

Assignment No. 2

Title

Assignment based on Classification

Problem Definition:
Classify the email using the binary classification method. Email Spam detection has two states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze their performance.
.

Prerequisite:

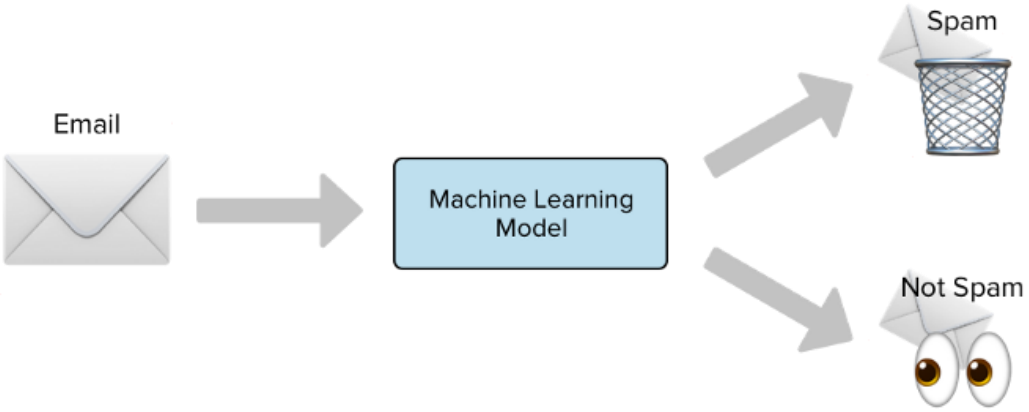
Basic of Python, Data Mining Algorithm, Concept of Classification

Software Requirements:

Anaconda with Python 3.7

Theory Concepts:

Motivation



Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

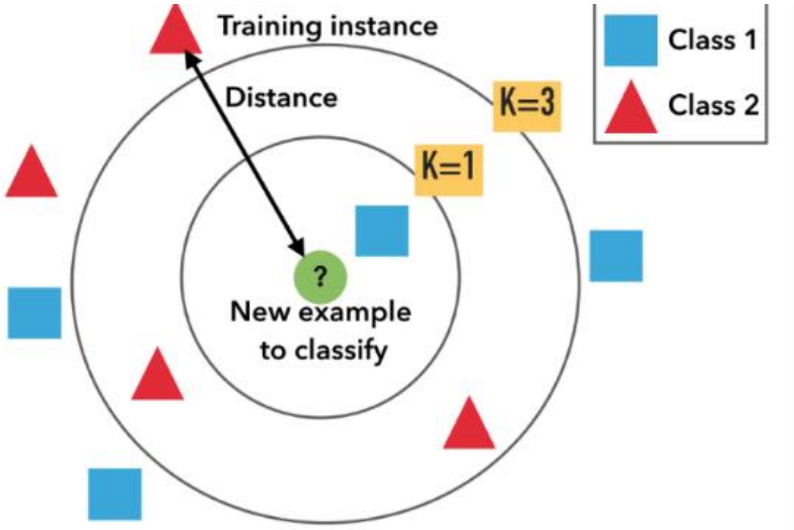
KNN is an *non parametric lazy learning* algorithm. That is a pretty concise statement. When you say a technique is non parametric , it means that it does not make any assumptions on the underlying data

distribution. This is pretty useful , as in the real world , most of the practical data does not obey the typical theoretical assumptions made (eg gaussian mixtures, linearly separable etc) . Non parametric algorithms like KNN come to the rescue here.

It is also a lazy algorithm. What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase* or it is very minimal. This means the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. More exactly, all the training data is needed during the testing phase. (Well this is an exaggeration, but not far from truth). This is in contrast to other techniques like SVM where you can discard all non support vectors without any problem. Most of the lazy algorithms – especially KNN – makes decision based on the entire training data set (in the best case a subset of them).

The dichotomy is pretty obvious here – There is a non existent or minimal training phase but a costly testing phase. The cost is in terms of both time and memory. More time might be needed as in the worst case, all data points might take point in decision. More memory is needed as we need to store all training data.

KNN Algorithm is based on feature similarity: How closely out-of-sample features resemble our training set determines how we classify a given data point:



Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).

KNN can be used for **classification**—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for **regression**—output is the

value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

Conclusion

In this way we and analysed Classified the email using the binary classification method.

Assignment Questions

1. Why does using SVM and kNN?
2. What is different classification algorithms used in ML?
3. What are the supervised and unsupervised learning, lazy learning?
4. What is hard margin and soft margin?
5. What affect the decision boundary in SVM?

Assignment No. 4

Title

Assignment based on Gradient Descent

Problem Definition:

Implement Gradient Descent Algorithm to find the local minima of a function.
For example, find the local minima of the function $y=(x+3)^2$ starting from the point x=2 .

Prerequisite:

Basic of Python, Data Mining Algorithm, Concept of gradient descent

Software Requirements:

Anaconda with Python 3.7

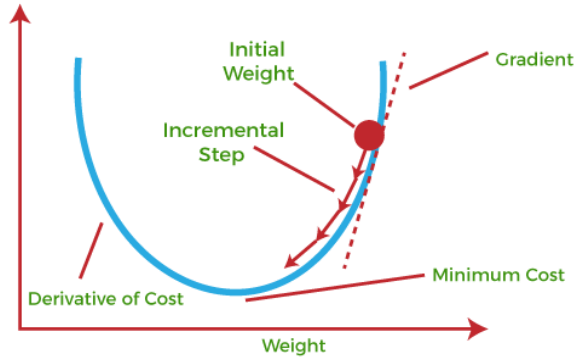
Theory Concepts:

Motivation

Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results.
Gradient Descent is defined as one of the most commonly used iterative optimization algorithms of machine learning to train the machine learning and deep learning models. It helps in finding the local minimum of a function.

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

- If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the **local minimum** of that function.
- Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the **local maximum** of that function.



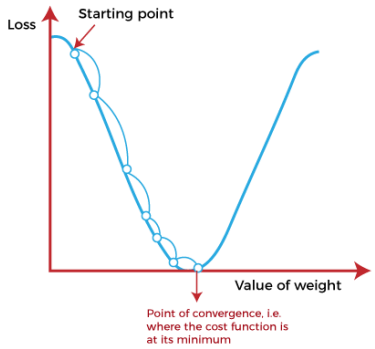
The main objective of using a gradient descent algorithm is to minimize the cost function using iteration. To achieve this goal, it performs two steps iteratively:

- Calculates the first-order derivative of the function to compute the gradient or slope of that function.
- Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

the working principle of gradient descent, The equation for simple linear regression is given as:

$$Y=mX+c$$

Where 'm' represents the slope of the line, and 'c' represents the intercepts on the y-axis.



The starting point(shown in above fig.) is used to evaluate the performance as it is considered just as an arbitrary point. At this starting point, we will derive the first derivative or slope and then use a tangent line to calculate the steepness of this slope. Further, this slope will inform the updates to the parameters (weights and bias).

The slope becomes steeper at the starting point or arbitrary point, but whenever new parameters are generated, then steepness gradually reduces, and at the lowest point, it approaches the lowest point, which is called **a point of convergence.**

The main objective of gradient descent is to minimize the cost function or the error between expected and actual. To minimize the cost function, two data points are required:

- **Direction**
- **Learning Rate**

These two factors are used to determine the partial derivative calculation of future iteration and allow it to the point of convergence or local minimum or global minimum

Conclusion

In this way we implemented gradient Descent Algorithms.

Assignment Questions

1. Why does gradient Descent used?
2. How to decide should using local minimum or local maximum?
3. What are the Different Application of gradient Descent?
4. What is Cost Function?
5. What is learning Rate?

Assignment No. 5

3.1 Title

Assignment based on k-NN Classification

3.2 Problem Definition:

Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

3.3 Prerequisite:

Basic of Python, Data Mining Algorithm, Concept of KNN Classification

3.4 Software Requirements:

Anaconda with Python 3.7

3.5 Theory Concepts:

3.5.1 Motivation

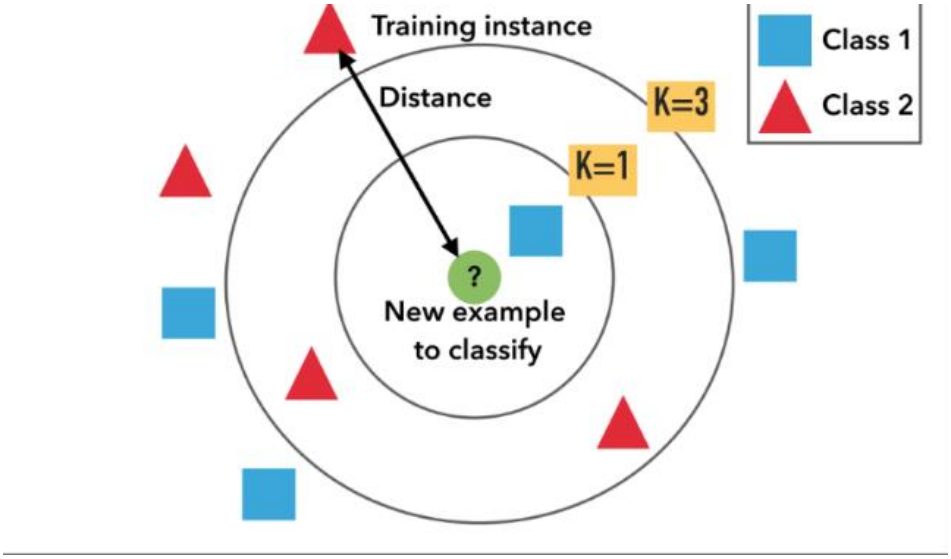
K-Nearest Neighbors (kNN) Algorithm-

KNN is an *non parametric lazy learning* algorithm. That is a pretty concise statement. When you say a technique is non parametric , it means that it does not make any assumptions on the underlying data distribution. This is pretty useful , as in the real world , most of the practical data does not obey the typical theoretical assumptions made (eg gaussian mixtures, linearly separable etc) . Non parametric algorithms like KNN come to the rescue here.

It is also a lazy algorithm. What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase* or it is very minimal. This means the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. More exactly, all the training data is needed during the testing phase. (Well this is an exaggeration, but not far from truth). This is in contrast to other techniques like SVM where you can discard all non support vectors without any problem. Most of the lazy algorithms – especially KNN – makes decision based on the entire training data set (in the best case a subset of them).

The dichotomy is pretty obvious here – There is a non existent or minimal training phase but a costly testing phase. The cost is in terms of both time and memory. More time might be needed as in the worst case, all data points might take point in decision. More memory is needed as we need to store all training data.

KNN Algorithm is based on **feature similarity**: How closely out-of-sample features resemble our training set determines how we classify a given data point:



Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).

KNN can be used for **classification**—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for **regression**—output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

Assumptions in KNN

Before using KNN, let us revisit some of the assumptions in KNN.

KNN assumes that the data is in a *feature space*. More exactly, the data points are in a metric space. The data can be scalars or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance – This need not necessarily be Euclidean distance although it is the one commonly used.

Each of the training data consists of a set of vectors and class label associated with each vector. In the simplest case , it will be either + or – (for positive or negative classes). But KNN , can work equally well with arbitrary number of classes.

We are also given a single number "k" . This number decides how many neighbors (where neighbors is defined based on the distance metric) influence the classification. This is usually a odd number if the number of classes is 2. If k=1 , then the algorithm is simply called the nearest neighbor algorithm.

KNN for Classification

Lets see how to use KNN for classification. In this case, we are given some data points for training and also a new unlabelled data for testing. Our aim is to find the class label for the new point. The algorithm has different behavior based on k.

Case 1 : k = 1 or Nearest Neighbor Rule

This is the simplest scenario. Let x be the point to be labeled . Find the point closest to x . Let it be y. Now nearest neighbor rule asks to assign the label of y to x. This seems too simplistic and some times even counter intuitive. If you feel that this procedure will result a huge error , you are right – but there is a catch. This reasoning holds only when the number of data points is not very large.

If the number of data points is very large, then there is a very high chance that label of x and y are same. An example might help – Lets say you have a (potentially) biased coin. You toss it for 1 million time and you have got head 900,000 times. Then most likely your next call will be head. We can use a similar argument here.

Let me try an informal argument here - Assume all points are in a D dimensional plane . The number of points is reasonably large. This means that the density of the plane at any point is fairly high. In other words , within any subspace there is adequate number of points. Consider a point x in the subspace which also has a lot of neighbors. Now let y be the nearest neighbor. If x and y are sufficiently close, then we can assume that probability that x and y belong to same class is fairly same – Then by decision theory, x and y have the same class.

The book "Pattern Classification" by Duda and Hart has an excellent discussion about this Nearest Neighbor rule. One of their striking results is to obtain a fairly tight error bound to the Nearest Neighbor rule. The bound is

$$P^* \leq P \leq P^*\left(2 - \frac{c}{c-1}P^*\right)$$

Where P^* is the Bayes error rate, c is the number of classes and P is the error rate of Nearest Neighbor. The result is indeed very striking (atleast to me) because it says that if the number of points is fairly large then the error rate of Nearest Neighbor is less that twice the Bayes error rate. Pretty cool for a simple algorithm like KNN.

Case 2 : k = K or k-Nearest Neighbor Rule

This is a straightforward extension of 1NN. Basically what we do is that we try to find the k nearest neighbor and do a majority voting. Typically k is odd when the number of classes is 2. Lets say k = 5 and there are 3 instances of C1 and 2 instances of C2. In this case , KNN says that new point has to labeled as C1 as it forms the majority. We follow a similar argument when there are multiple classes.

One of the straight forward extension is not to give 1 vote to all the neighbors. A very common thing to do is *weighted kNN* where each point has a weight which is typically calculated using its distance. For eg under inverse distance weighting, each point has a weight equal to the inverse of its distance to the point to be classified. This means that neighboring points have a higher vote than the farther points.

It is quite obvious that the accuracy *might* increase when you increase k but the computation cost also increases.

Some Basic Observations

1. If we assume that the points are d-dimensional, then the straight forward implementation of finding k Nearest Neighbor takes O(dn) time.
2. We can think of KNN in two ways – One way is that KNN tries to estimate the posterior probability of the point to be labeled (and apply bayesian decision theory based on the posterior probability). An alternate way is that KNN calculates the decision surface (either implicitly or explicitly) and then uses it to decide on the class of the new points.
3. There are many possible ways to apply weights for KNN – One popular example is the Shephard's method.
4. Even though the naive method takes O(dn) time, it is very hard to do better unless we make some other assumptions. There are some efficient data structures like **KD-Tree** which can reduce the time complexity but they do it at the cost of increased training time and complexity.
5. In KNN, k is usually chosen as an odd number if the number of classes is 2.
6. Choice of k is very critical – A small value of k means that noise will have a higher influence on the result. A large value make it computationally expensive and kinda defeats the basic philosophy behind KNN (that points that are near might have similar densities or classes) .A simple approach to select k is set $k = \sqrt{n}$
7. There are some interesting data structures and algorithms when you apply KNN on graphs – See **Euclidean minimum spanning tree** and **Nearest neighbor graph** .

8. There are also some nice techniques like condensing, search tree and partial distance that try to reduce the time taken to find the k nearest neighbor.

Applications of KNN

KNN is a versatile algorithm and is used in a huge number of fields. Let us take a look at few uncommon and non trivial applications.

1. Nearest Neighbor based Content Retrieval

This is one the fascinating applications of KNN – Basically we can use it in Computer Vision for many cases – You can consider handwriting detection as a rudimentary nearest neighbor problem. The problem becomes more fascinating if the content is a video – given a video find the video closest to the query from the database – Although this looks abstract, it has lot of practical applications – Eg : Consider **ASL** (American Sign Language) . Here the communication is done using hand gestures.

So lets say if we want to prepare a dictionary for ASL so that user can query it doing a gesture. Now the problem reduces to find the (possibly k) closest gesture(s) stored in the database and show to user. In its heart it is nothing but a KNN problem. One of the professors from my dept , Vassilis Athitsos , does research in this interesting topic – See **Nearest Neighbor Retrieval and Classification** for more details.

2. Gene Expression

This is another cool area where many a time, KNN performs better than other state of the art techniques . In fact a combination of KNN-SVM is one of the most popular techniques there. This is a huge topic on its own and hence I will refrain from talking much more about it.

3. Protein-Protein interaction and 3D structure prediction

Graph based KNN is used in protein interaction prediction. Similarly KNN is used in structure prediction.

4. Credit ratings—collecting financial characteristics vs. comparing people with similar financial features to a database. By the very nature of a credit rating, people who have similar financial details would be given similar credit ratings. Therefore, they would like to be able to use this existing database to predict a new customer's credit rating, without having to perform all the calculations.

5. Should the bank give a loan to an individual? Would an individual default on his or her loan? Is that person closer in characteristics to people who defaulted or did not default on their loans?

6. In political science—classing a potential voter to a "will vote" or "will not vote", or to

"vote Democrat" or "vote Republican".

7. More advance examples could include handwriting detection (like OCR), image recognition and even video recognition.

Pros:

- No assumptions about data—useful, for example, for nonlinear data
- Simple algorithm—to explain and understand/interpret
- High accuracy (relatively)—it is pretty high but not competitive in comparison to better supervised learning models
- Versatile—useful for classification or regression

Cons:

- Computationally expensive—because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data

3.6 Algorithm

1. Import the Required Packages
2. Read Given Dataset
3. Import KNeighborshood Classifier and create object of it.
4. Predict the class for the point(6,6) w.r.t to General KNN.
5. Predict the class for the point(6,6) w.r.t to Distance Weighted KNN.

3.7 Conclusion

In this way we learn KNN Classification to predict the General and Distance Weighted KNN for Given data point in term of Positive or Negative.

3.12 Assignment Questions

1. How KNN Different from Kmean Clustering Algorithm?
2. What is the Formula for Euclidean Distance?
3. What is Hamming Distance?
4. What is formula for Manhatten and Minkowski Distance?
5. What are the application of KNN?
6. Explain confusion matrix. Recall,

Assignment No. 6

Title

Assignment based on k-mean Clustering

Problem Definition:

Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

Prerequisite:

Basic of Python, Data Mining Algorithm, Concept of K-mean Clustering

Software Requirements:

Anaconda with Python 37

Theory Concepts:

Motivation

A Hospital Care chain wants to open a series of Emergency-Care wards within a region. We assume that the hospital knows the location of all the maximum accident-prone areas in the region. They have to decide the number of the Emergency Units to be opened and the location of these Emergency Units, so that all the accident-prone areas are covered in the vicinity of these Emergency Units.

The challenge is to decide the location of these Emergency Units so that the whole region is covered. Here is when K-means Clustering comes to rescue!

A cluster refers to a small group of objects. Clustering is grouping those objects into clusters. In order to learn clustering, it is important to understand the scenarios that lead to cluster different objects. Let us identify a few of them.

What is Clustering?

Clustering is dividing data points into homogeneous classes or clusters:

Points in the same group are as similar as possible

Points in different group are as dissimilar as possible

When a collection of objects is given, we put objects into group based on similarity.

Application of Clustering:

Clustering is used in almost all the fields. You can infer some ideas from Example 1 to come up with lot of clustering applications that you would have come across.

Listed here are few more applications, which would add to what you have learnt.

- ✓ Clustering helps marketers improve their customer base and work on the target areas. It helps group people (according to different criteria's such as willingness, purchasing power etc.) based on their similarity in many ways related to the product under

consideration.

- ✓ Clustering helps in identification of groups of houses on the basis of their value, type and geographical locations.
- ✓ Clustering is used to study earth-quake. Based on the areas hit by an earthquake in a region, clustering can help analyse the next probable location where earthquake can occur.

Clustering Algorithms:

A Clustering Algorithm tries to analyse natural groups of data on the basis of some similarity. It locates the centroid of the group of data points. To carry out effective clustering, the algorithm evaluates the distance between each point from the centroid of the cluster.
The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data.

What is K-means Clustering?

K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.

K-means Clustering – Example 1:

A pizza chain wants to open its delivery centres across a city. What do you think would be the possible challenges?

- They need to analyse the areas from where the pizza is being ordered frequently.
- They need to understand as to how many pizza stores has to be opened to cover delivery in the area.
- They need to figure out the locations for the pizza stores within all these areas in order to keep the distance between the store and delivery points minimum.

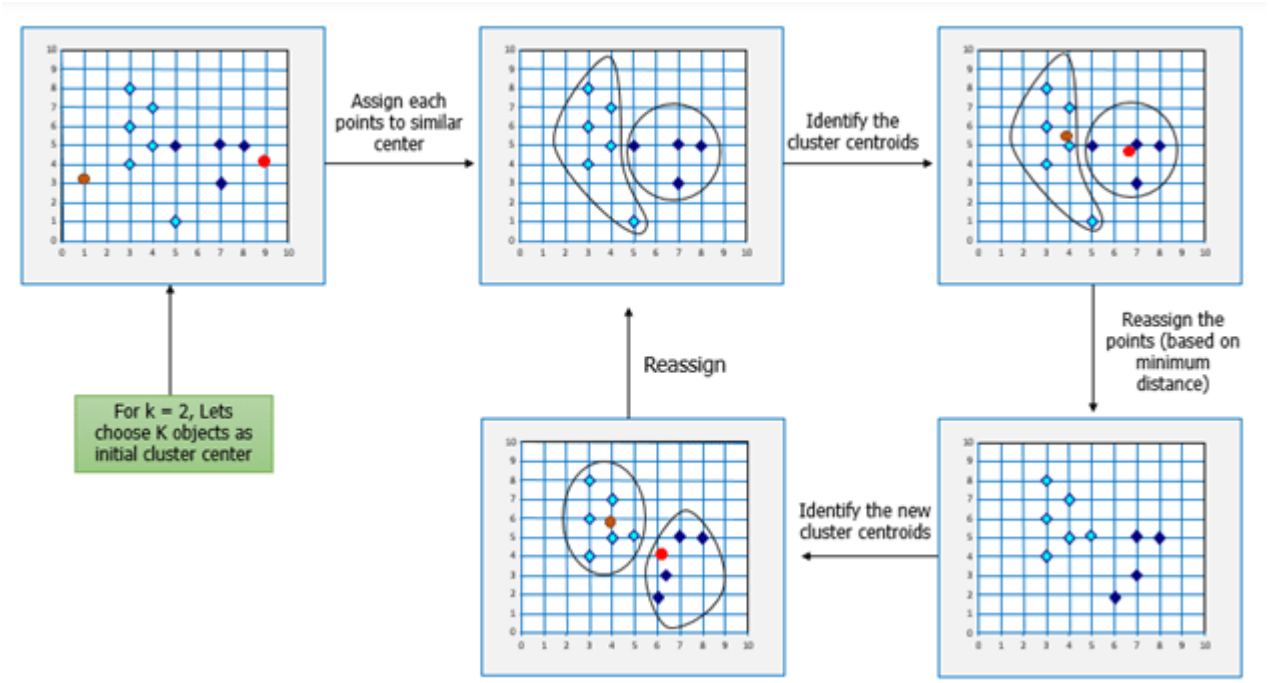
Resolving these challenges includes a lot of analysis and mathematics. We would now learn about how clustering can provide a meaningful and easy method of sorting out such real life challenges. Before that let's see what clustering is.

K-means Clustering Method:

If k is given, the K-means algorithm can be executed in the following steps:

- Partition of objects into k non-empty subsets
- Identifying the cluster centroids (mean point) of the current partition.
- Assigning each point to a specific cluster
- Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.
- After re-allotting the points, find the centroid of the new cluster formed.

The step by step process:



Now, let's consider the problem in Example 1 and see how we can help the pizza chain to come up with centres based on K-means algorithm.

Similarly, for opening Hospital Care Wards:
K-means Clustering will group these locations of maximum prone areas into clusters and define a cluster center for each cluster, which will be the locations where the Emergency Units will open. These Clusters centers are the centroids of each cluster and are at a minimum distance from all the points of a particular cluster, henceforth, the Emergency Units will be at minimum distance from all the accident prone areas within a cluster.
Here is another example for you, try and come up with the solution based on your understanding of K-means clustering.

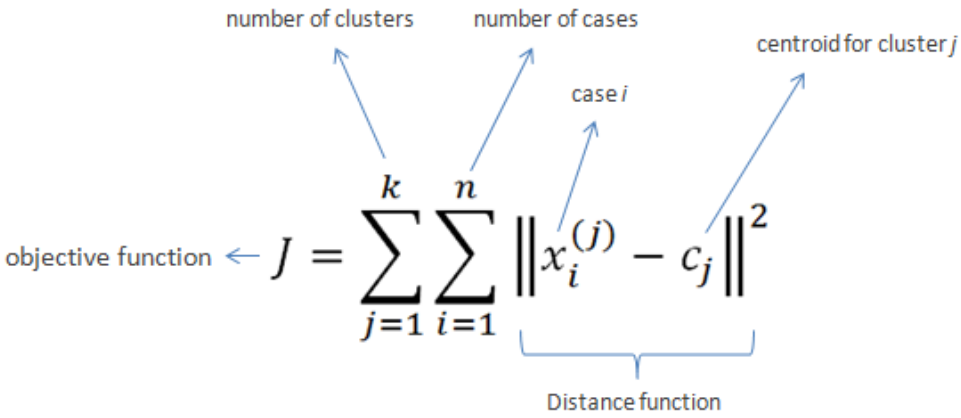
K-means Clustering – Example 2:
Let's consider the data on drug-related crimes in Canada. The data consists of crimes due to various drugs that include, Heroin, Cocaine to prescription drugs, especially by underage people. The crimes resulted due to these substance abuse can be brought down by starting de-addiction centres in areas most afflicted by this kind of crime. With the available data, different objectives can be set. They are:

- Classify the crimes based on the abuse substance to detect prominent cause.
- Classify the crimes based on age groups.
- Analyze the data to determine what kinds of de-addiction centre is required.
- Find out how many de-addiction centres need to be setup to reduce drug related crime rate.

The K-means algorithm can be used to determine any of the above scenarios by analyzing the available data.
Following the K-means Clustering method used in the previous example, we can start off with a given k, following by the execution of the K-means algorithm.

Mathematical Formulation for K-means Algorithm:

K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the clust nearest mean. This method produces exactly *k* different clusters of greatest possible distinction. The best clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed fro objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function



Conclusion

In this way we learn Kmean Clustering Algorithm using elbow methods.

Assignment Questions

1. Why does k-means clustering algorithm use only Euclidean distance metric?
2. How to decide on the correct number of clusters?
3. What are the Different Application of K Mean Clustering?
4. What is K-medoids?
5. What is *k*-medians clustering?