# LAB 12: K-NEAREST NEIGHBORS & K-MEANS

## What is Machine learning

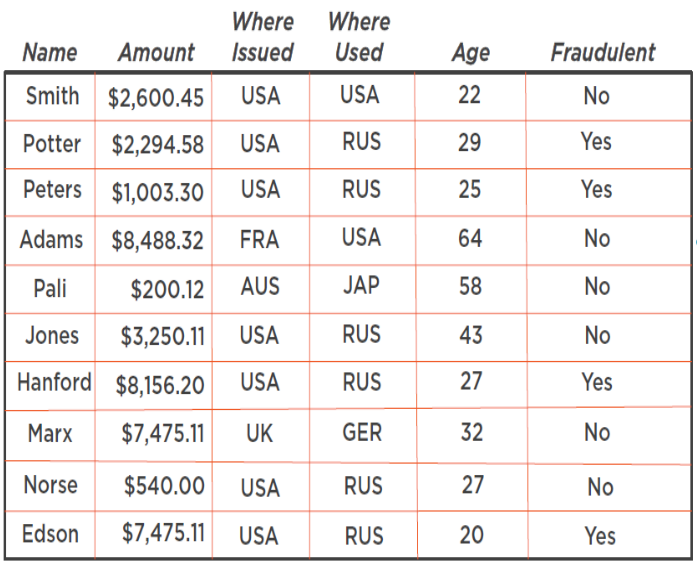
There's probably no definition that the whole world would agree on, but there certainly are some core concepts. To think about those, think about what machine learning does? Machine learning finds patterns in data. And Uses those patterns to predict the future
Examples:

- Detecting credit card fraud
- Determining whether a customer is likely to switch to a competitor
- Deciding when to do preventive maintenance on a factory robot

The core thing machine learning does is finds **patterns** in data. It then uses those patterns to predict the **future**. For example, you can use machine learning to detect credit card fraud. Suppose you have data about previous credit card transactions. You could find patterns in that data, potentially, that will let you detect when a new credit card transaction is likely to be fraudulent, or maybe you want to determine whether a customer is likely to switch to a competitor. The core idea is that machine learning lets you find patterns in data, then use those patterns to predict the future.

## Finding Patterns

**What does it mean to learn? For example, how did you learn to read?**
Well, learning requires identifying patterns. In reading, for instance, you identify letters and then the patterns of letters together to form words. You then had to recognize those patterns when you saw them again. That's what learning means, just as when you learn to read. And that's what machine learning does with data that we provide. So, for example, suppose I have data about credit card transactions. Suppose I have following data.



| Name | Amount | Where Issued | Where Used | Age | Fraudulent |
|---|---|---|---|---|---|
| Smith | $2,600.45 | USA | USA | 22 | No |
| Potter | $2,294.58 | USA | RUS | 29 | Yes |
| Peters | $1,003.30 | USA | RUS | 25 | Yes |
| Adams | $8,488.32 | FRA | USA | 64 | No |
| Pali | $200.12 | AUS | JAP | 58 | No |
| Jones | $3,250.11 | USA | RUS | 43 | No |
| Hanford | $8,156.20 | USA | RUS | 27 | Yes |
| Marx | $7,475.11 | UK | GER | 32 | No |
| Norse | $540.00 | USA | RUS | 27 | No |
| Edson | $7,475.11 | USA | RUS | 20 | Yes |

In this data I know where the card was issued, where it was used, the age of the user. Now, what's the pattern for fraudulent transactions? Well, it turns out that if you look at that, there really is a pattern in this data. It is that a transaction is fraudulent if the card holder is in their 20's, if the card is issued the USA and used in Russia, and the amount is more than $1000.

## K Nearest Neighbors

The purpose of the k Nearest Neighbors (kNN) algorithm is to use a database in which the data points are separated into several separate classes to predict the classification of a new sample point. Let's look at an example to understand it.

Suppose a botanist wants to study the diversity of flowers growing in a large meadow. However, he does not have time to examine each flower himself and cannot afford to hire assistants who know about flowers to help him. Instead, he gets people to measure various characteristics of the flowers, such as the stamen size, the number of petals, the height, the color, the size of the flower head, etc, and put them into a computer. He then wants the computer to compare them to a pre-classified database of samples, and predict what variety each of the flowers is, based on its characteristics.

In general, we start with a set of data, each data point of which is in a known **class**. Then, we want to be able to **predict** the classification of a new data point based on the known classifications of the observations in the database. For this reason, the database is known as our **training set**, since it trains us what objects of the different classes look like. The process of choosing the classification of the new observation is known as the classification problem, and there are several ways to tackle it. Here, we consider choosing the classification of the new observation based on the classifications of the observations in the database which it is "most similar" to.

### Defining Similarity

However, deciding whether two observations are similar or not is quite an open question. For instance, deciding whether two colours are similar is a completely different process to deciding whether two paragraphs of text are similar. Clearly, then, before we can decide whether two observations are similar, we need to find some way of comparing objects. The principle trouble with this is that our data could be of many different types - it could be a number, it could be a colour, it could be a geographical location, it could be a true/false (boolean) answer to a question, etc - which would all require different ways of measuring similarity.

## Data Pre-Processing

It seems then that this first problem is one of preprocessing the data in the database in such a way as to ensure that we can compare observations. One common way of doing this is to try to convert all our characteristics into a numerical value, such as converting colours to RGB values, converting locations to latitude and longitude, or converting boolean values into ones and zeros. Once we have everything as numbers, we can imagine a space in which each of our characteristics is represented by a different dimension, and the value of each observation for each characteristic is its coordinate in that dimension.

Then, our observations become points in space and we can interpret the distance between them as their similarity (using some appropriate metric). Even once we have decided on some way of determining how similar two observations are, we still have the problem of deciding which observations from the database are similar enough to our new observation for us to take their classification into account when classifying the new observation. This problem can be solved in several different ways, either by considering all the data points within a certain radius of the new sample point, or by taking only a certain number of the nearest points. This latter method is what we consider now in the k Nearest Neighbours Algorithm

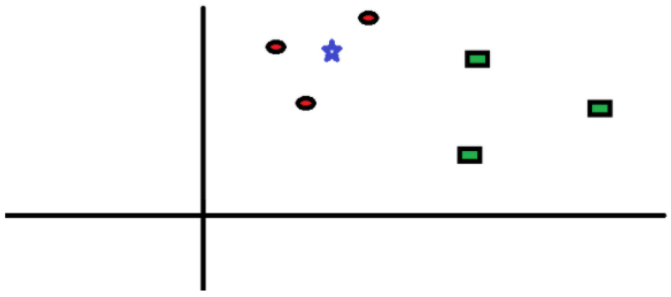
## Algorithm

The Algorithm The algorithm can be summarized as:

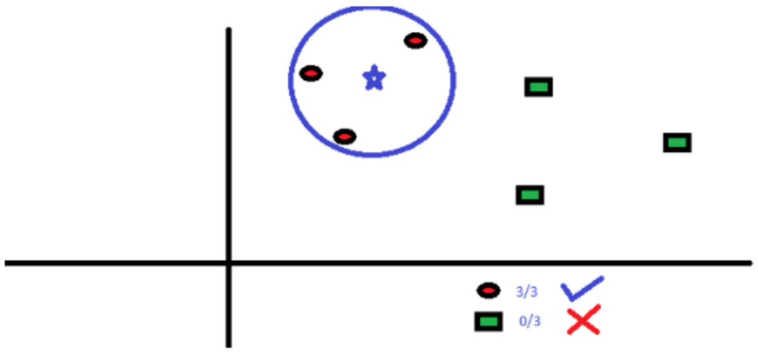
1. A positive integer k is specified, along with a new sample
2. We select the k entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample

## How does the KNN algorithm work?

Let’s take a simple case to unders tand this algorithm. Following is a spread of red circles (RC) and green squares (GS) :



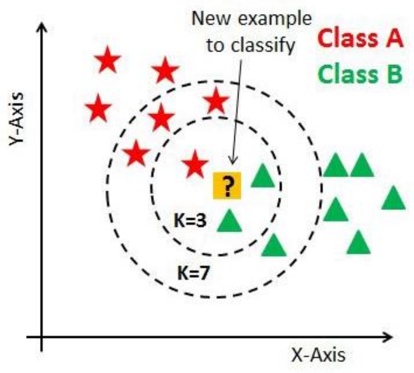
You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The “K” is KNN algorithm is the nearest neighbor we wish to take the vote from. Let’s say K = 3. Hence, we will now make a circle with BS as the center just as big as to enclose only three datapoints on the plane. Refer to the following diagram for more details:



The three closest points to BS is all RC. Hence, with a good confidence level, we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm. Next, we will understand what the factors are to be considered to conclude the best K.

## How do we choose the factor K?

K value indicates the count of the nearest neighbors. We have to compute distances between test points and trained labels points. Updating distance metrics with every iteration is computationally expensive, and that’s why KNN is a lazy learning algorithm.



- As you can verify from the above image, if we proceed with K=3, then we predict that test input belongs to class B, and if we continue with K=7, then we predict that test input belongs to class A.
- That's how you can imagine that the K value has a powerful effect on KNN performance.

Then how to select the optimal K value?

- There are no pre-defined statistical methods to find the most favourable value of K.
- Initialize a random K value and start computing.
- Choosing a small value of K leads to unstable decision boundaries.
- The substantial K value is better for classification as it leads to smoothening the decision boundaries.
- **Derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate.**

Let's now implement the KNN model to see what we have studied in action.

## Datasets

We have a dataset having some health attributes of a patients along with the information that whether the patient has diabetes or not.Find the provided csv file named "**diabetes.csv**".

Lets look at the dataset and see what data we have. The file contains following attributes

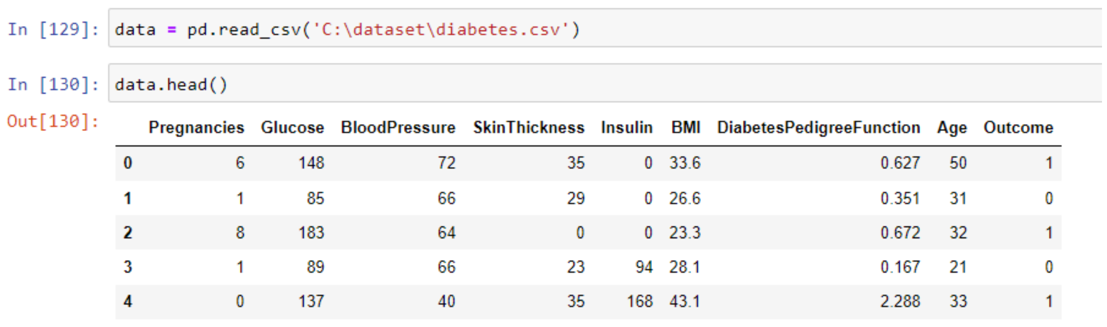
1. Pregnancies
2. Glucose
3. Blood Pressure
4. Skin Thickness
5. Insulin
6. BMI
7. Diabetes
8. Pedigree Function

9. Age
10. Outcome

## Task 1:

Use diabetes.csv for this task.

1. Load the dataset in Python



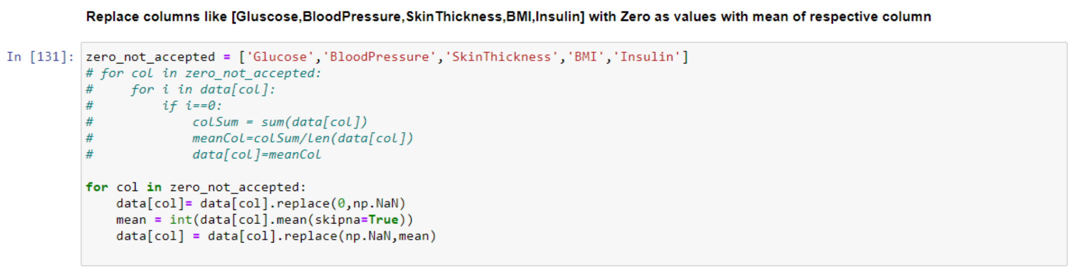
```
In [129]: data = pd.read_csv('C:\dataset\diabetes.csv')

In [130]: data.head()

Out[130]:
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

2. Clean the data
   Replace the missing values in the dataset with mean value.

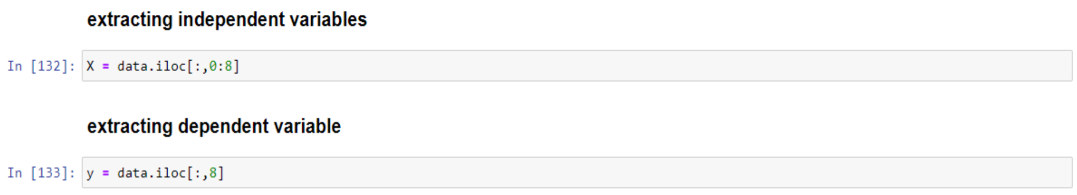


**Replace columns like [Gluscose,BloodPressure,SkinThickness,BMI,Insulin] with Zero as values with mean of respective column**

```
In [131]: zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']
          # for col in zero_not_accepted:
          #     for i in data[col]:
          #         if i==0:
          #             colSum = sum(data[col])
          #             meanCol=colSum/len(data[col])
          #             data[col]=meanCol

          for col in zero_not_accepted:
              data[col]= data[col].replace(0,np.NaN)
              mean = int(data[col].mean(skipna=True))
              data[col] = data[col].replace(np.NaN,mean)
```

3. Separate the dependent and independent variable.



**extracting independent variables**

```
In [132]: X = data.iloc[:,0:8]
```

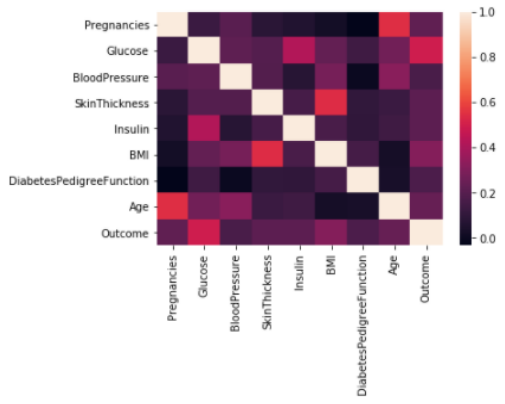
**extracting dependent variable**

```
In [133]: y = data.iloc[:,8]
```

4. Explore the data

**Exploring data to know relation before processing**

```
In [134]: sns.heatmap(data.corr())
Out[134]: <matplotlib.axes._subplots.AxesSubplot at 0x1b60d430>
```



5. Split the data into training and testing sets.

**splitting dataset into training and testing set**

```
In [86]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

6. Now it’s your turn. We have prepared our data. Now the next step is to implement the KNN algorithm
   1. Create a function named **KNNClassifier** that take input the value of K.
   2. Use **Euclidean Distance** measure to find the distance between two datapoints. (Create a separate function that will take two instances and return the distance)
   3. Create a function **Predict** and make predictions using the function we created for KNN classification. Call the function with different values of the K and print actual and predicted values.

## K-Mean Clustering

One of the most popular machine learning algorithms to perform clustering which allows us to maximize inter-cluster similarity and minimize intra-cluster similarity is the K-means clustering algorithm.
Let's say we have a number of points in two-dimensional space. This can be extended to N dimensional space. We'll work with two dimensions because that's simpler to visualize. We start off by initializing K centroids or the K-means of the clusters.

In K-means clustering you have to specify this value of K up front, how many clusters you want
your data to be divided into. Let's assume 4 different centroids in our example.

Once you have K cluster centers assign each point to a particular cluster. In order to do this, we calculate the distance between every point and every cluster center. A point is assigned to
that cluster whose cluster center it is the closest to.

Once you've assigned all the points, you'll see a cluster set up like this. At this point in time, use the existing points in each cluster to recalculate the mean for each cluster. Once the cluster centers have been recalculated; you'll find that certain points will move to another cluster.

We recalculate the distance from all cluster centers and reassign the points. This process of recalculating the means of each cluster and then reassigning the points once the new means have been calculated continues till the points reach their final position. When the cluster centers and the corresponding points don't move anymore, that's when the algorithm has converged. After convergence, you can think of every cluster being represented by a single point and this point is the reference vector. This reference vector is the center of the cluster and because it is calculated as an average of all points that belong to a cluster it's called the centroid of the cluster.

## Task 2:

Use dataset2.csv for this task.
You have to apply k means clustering algorithm on the provided dataset. Divide the data into 10 clusters and display those 10 clusters in a separate Data frame.

**Note:** You have to write algorithm of k-means clustering using only NumPy and pandas extract predicted clusters out of it. You are not allowed to use any library.