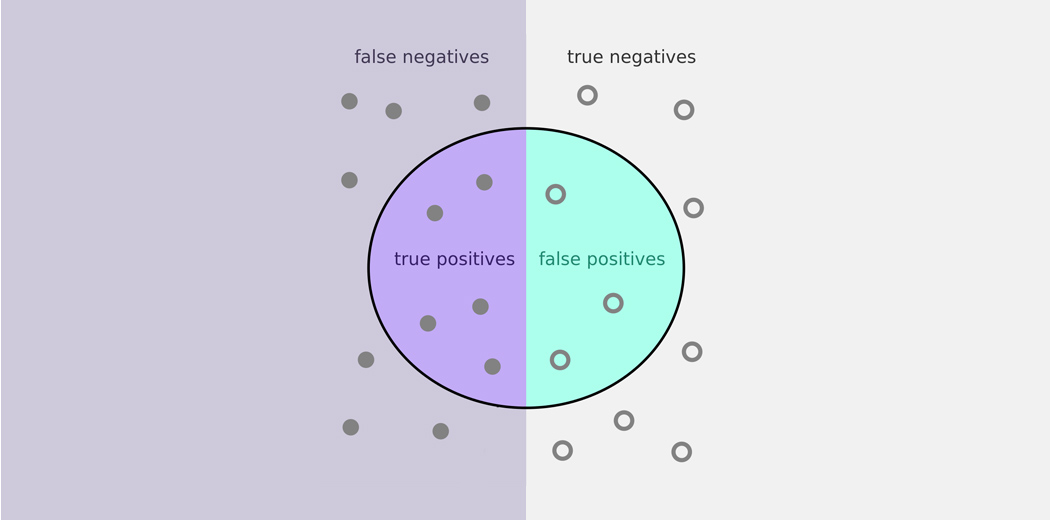
**What is precision and recall?**

As a machine learning professor or data scientist the most confusing part in there learning journey is the difference between precision and recall

Actually, the difference between precision and recall is easy to remember provided you have a better understanding of it like what each term actually means.



Source: <https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/>

To make any machine learning model, we all know to achieve the good fit model is very important and very challenging at times. This also involves achieving balance between overfitting and underfitting of the model or in more clear way a trade-off between bias and variance.

However, when talking about classification there comes another trade-off that often overlooked in favour of bias-variance trade-off which is for precision-recall trade-off. Imbalanced classes occur commonly in datasets and when it comes to specific use cases, we would in fact like to give more importance to the precision and recall metrics, and also how to achieve the balance between them.

We will use KNN model to simplify prediction

It is always good to import necessary library

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

from sklearn.metrics import roc\_curve

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import precision\_recall\_curve

from sklearn.metrics import auc

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

Now next look at the data and the target variable

data\_file\_path = '..pathToDirectory/input/heart-disease-uci/heart.csv'

data\_df = pd.read\_csv(data\_file\_path)

data\_df.head()//to know the data information

have to use EDA techniques to remove outlier and other necessary data

than we have to split the test and train data

y = data\_df["target"].values

x = data\_df.drop(["target"], axis = 1)

#Scaling - mandatory for knn

from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

x = ss.fit\_transform(x)

#Splitting into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3) # 70% training and 30% test

The reason we are choosing the best value of k is to save the time of trial and error, always keep in mind that we can determine the optimum value of k when we get the highest test score for that value. For that, we can evaluate the training and testing scores for up to 20 nearest neighbours:

train\_score = []

test\_score = []

k\_vals = []

for k in range(1, 21):

k\_vals.append(k)

knn = KNeighborsClassifier(n\_neighbors = k)

knn.fit(X\_train, y\_train)

tr\_score = knn.score(X\_train, y\_train)

train\_score.append(tr\_score)

te\_score = knn.score(X\_test, y\_test)

test\_score.append(te\_score)

below code can be use for to evaluate max score

## score that comes from the testing set only

max\_test\_score = max(test\_score)

test\_scores\_ind = [i for i, v in enumerate(test\_score) if v == max\_test\_score]

print('Max test score {} and k = {}'.format(max\_test\_score \* 100, list(map(lambda x: x + 1, test\_scores\_ind

Thus, we have obtained the optimum value of k to be 3, 11, or 20 with a score of 83.5. We will finalize one of these values and fit the model accordingly:

Knn classifier can be identify as

#Setup a knn classifier with k neighbors

knn = KNeighborsClassifier(3)

knn.fit(X\_train, y\_train)

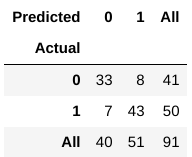
knn.score(X\_test, y\_test)

now we have to identify confusion matrix

y\_pred = knn.predict(X\_test)

confusion\_matrix(y\_test,y\_pred)

pd.crosstab(y\_test, y\_pred, rownames = ['Actual'], colnames =['Predicted'], margins = True)



A confusion matrix helps us gain an insight into how correct our predictions were and how they hold up against the actual values.

From our train and test data, we already know that our test data consisted of 91 data points. That is the 3rd row and 3rd column value at the end. We also notice that there are some actual and predicted values. The actual values are the number of data points that were originally categorized into 0 or 1. The predicted values are the number of data points our KNN model predicted as 0 or 1.

The actual values are:

* The patients who actually don’t have a heart disease = 41
* The patients who actually do have a heart disease = 50

The predicted values are:

* Number of patients who were predicted as not having a heart disease = 40
* Number of patients who were predicted as having a heart disease = 51

All the values we obtain above have a term. Let’s go over them one by one:

* The cases in which the patients actually did not have heart disease and our model also predicted as not having it is called the **True Negatives.** For our matrix, True Negatives = 33.
* The cases in which the patients actually have heart disease and our model also predicted as having it are called the **True Positives.** For our matrix, True Positives = 43
* However, there are some cases where the patient actually no heart disease has, but our model has predicted that they do. This kind of error is the Type I Error and we call the values as **False Positives.** For our matrix, False Positives = 8
* Similarly, there are some cases where the patient actually heart disease has, but our model has predicted that he/she don’t. This kind of error is the Type II Error and we call the values as **False Negatives.**  For our matrix, False Negatives = 7

**What is Precision?**

Right – so now we come to the crux of this article. What in the world is Precision? And what does all the above learning have to do with it?

In the simplest terms, Precision is the ratio between the True Positives and all the Positives. For our problem statement, that would be the measure of patients that we correctly identify having a heart disease out of all the patients actually having it. Mathematically:

Precision and Recall - Precision

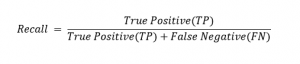
Source: <https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/>

What is the Precision for our model? Yes, it is 0.843 or, when it predicts that a patient has heart disease, it is correct around 84% of the time.

Precision also gives us a measure of the relevant data points. It is important that we don’t start treating a patient who actually doesn’t have a heart ailment, but our model predicted as having it.

**What is Recall?**

The recall is the measure of our model correctly identifying True Positives. Thus, for all the patients who actually have heart disease, recall tells us how many we correctly identified as having a heart disease. Mathematically:



Source: <https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/>

For our model, Recall  = 0.86. Recall also gives a measure of how accurately our model is able to identify the relevant data. We refer to it as Sensitivity or True Positive Rate. What if a patient has heart disease, but there is no treatment given to him/her because our model predicted so? That is a situation we would like to avoid!