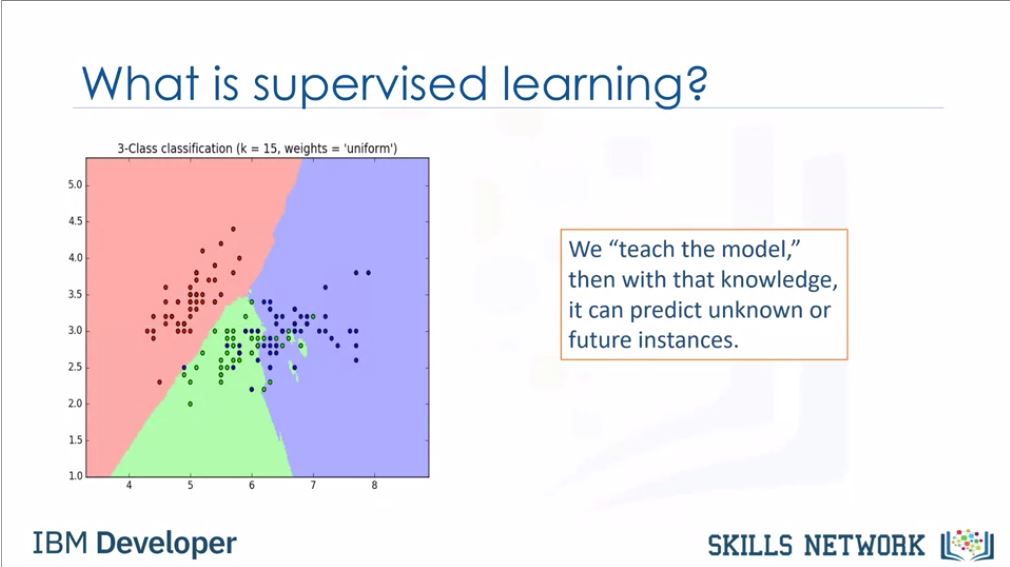
**Classification:**

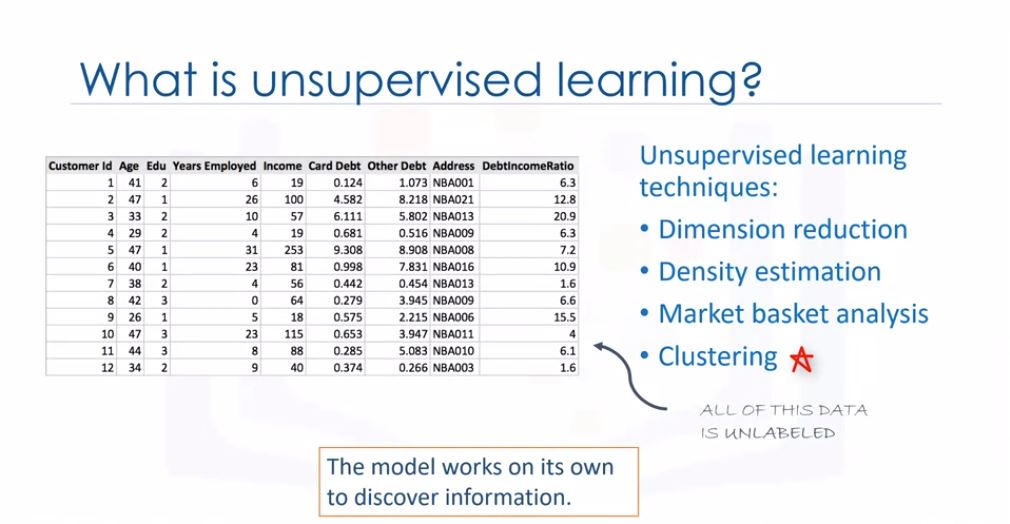
- A supervised approach.

- Categorizing or classifying the unknown items to discrete set of categories or classes

- The target attribute is categorical value.



It determines the class label for an unlabeled test case.



Suppose we have data of the loan takers. Now a person want to take a loan .

So he/she will provide the information like the income , age etc. So from that we can

predict whether he/she will repay the loan or loan should be cancelled.

Binary classifier - Same as above example . If person is defaulter then value is 1

else value of default is 0.

We can also built multi - class classification.

Multi class classification example- If the data of patients with same health problems

are there tested with one of the three drugs. Now we also have data that which drug

cured them. So for unknown patient with same health problems can be helped by

predicting which from the three drugs should be used for treatment.

Bussiness uses:

- customer category

- will customer switches to another provider or not?

- will the customer respond to the advertising or not?

Others:

- email filtering

- image recognition

- etc.

TYPES:

- Decision trees

-Naive bayes

-Linear discriminant Analysis

-k-nearest neighbour

-Logistic regression

-Nueral networks

-Support vector machines(SVM)

**KNN:**

Problem: Telecommunication company wants to categorised the customer on the basis

of their usage . Four pre defined labels are used to define the user:

1.Basic service

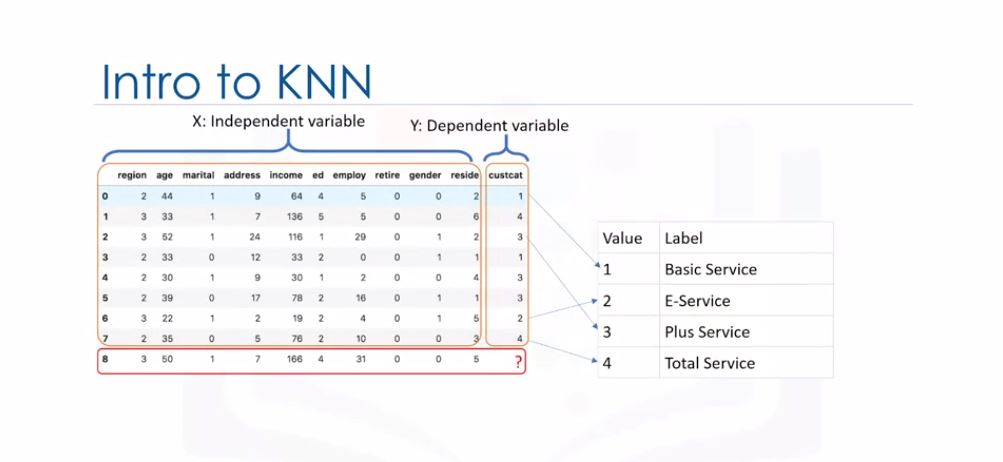
2.E-service

3.Plus service

4.Total service

Using this categorisation the marketing to the targeted customer can be done easily.

Objective : Make a classifier.



TOTAL FIELDS:

region,age,martial,address,income,ed,employee,retire,gender,reside,custcat

Here if we consider 2 factors age and income.

Here we have past datasets with categorisation done.

Now we enter the values of income and age of a new customer.Now we have to predict the

category for that new customer. So how can we predict? Using the K-nearest neighbour

we can plot the graph of age and income with new value and previous datasets.



And the category of the nearest data to the new value can be assigned and can be predicted

that the new customer will be most likely go in this category. The nearest is called the

1st KNN i.e. 1NN.

Now upto what extent can we trust the 1NN? If we have the first nieghbour as the exception or outlier

then our prediction can be called wrong!

Hence rather than choosing the 1 KNN we select 5 neighbours. Example if we select

5 NN and 3 NN goes to the category 3 i.e. plus service. So possibility of the category

for the new customer is 3rd category.

Here we took 5 nearest neighbour so in KNN value of K is 5.

What is KNN?

A method for classifying cases based on the similiarity to other cases.

Cases that are near each other are said to be neighbors.

Based on similiar cases with same class labels are near each other.

There distance shows the dissimiliarity which can be found by many ways like

Euclidian distances.

Algorithm works:

1.Pick a value for K

2.Calculate the distance of unknown case from all cases.

3.Select the K observations in data that are nearest to unknown one.

4.Predict the response of the unknown data point using the most popular response.

How to calculate similiarity of datapoints:

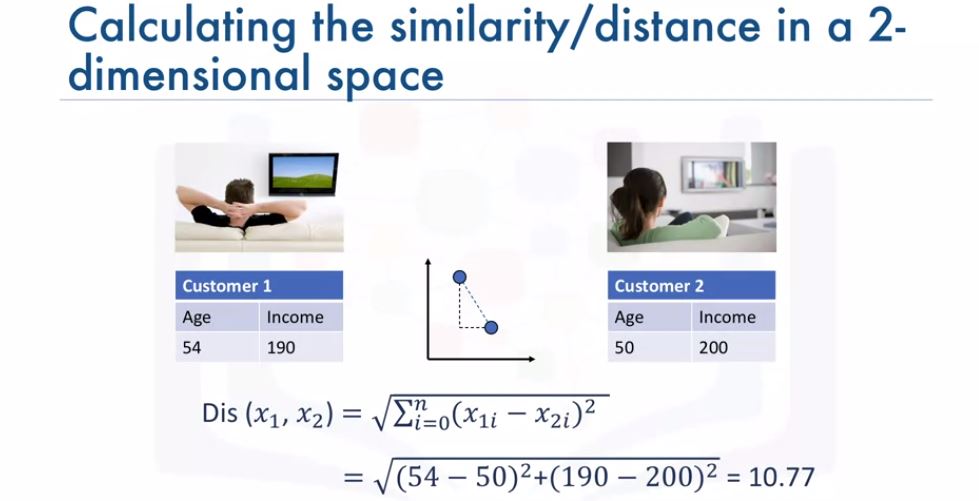
If only age:

Age : cust1:54 cust2:50

To find distance/similiarity use Euclidian distance : \sqrt(54-50)^2=4

If both income and age

Distance \sqrt{(x1-x)^2+(x2-x)^2}



Value of K:

K=1 : class 1

low K sets highly complex model might result in overfitting of the model

Overfitting is bad.

High K makes model overly generalized.

So how to select k:

1. Select K=1 and then make the model and pass the test data to find the accuracy.

2. Now increase the k linearly and repeat the process.

The one value of K with high accuracy of prediction is to be selected.

**Evaluation metrics explains the performance of the model.**

So let’s take an example. Suppose we are predicting the customer churn for the telecommunication

company. So we first use the data set or train set to train the model and then we

pass the train set to to predict the values of the churn of the customer.

To calculate the accuracy of the model we compare the predicted values of the labels with the

actual values values of the labels.

So evaluation metrics is used to calculate the acccuracy and provide the insights

of the model to improve it or not.

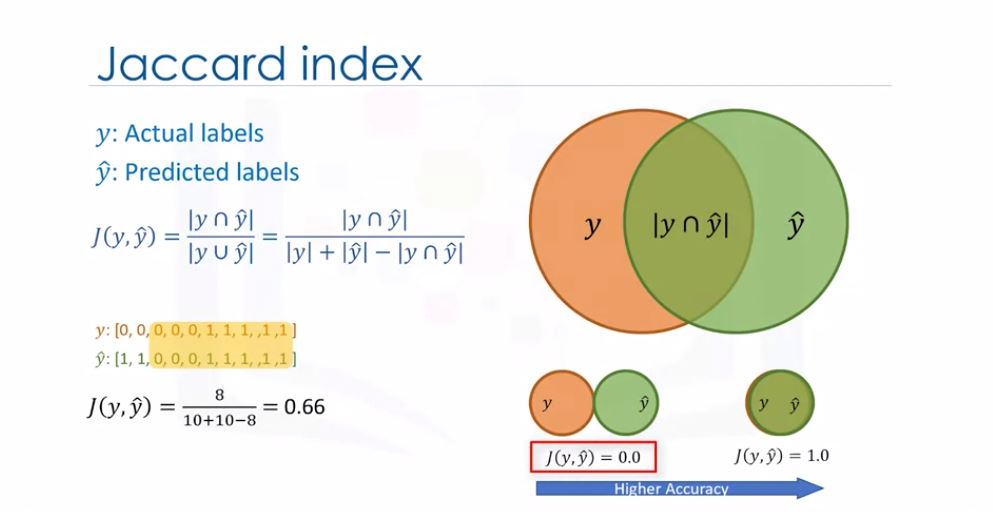
Types of evaluation metrics:Jaccard index ,F1 score , Log loss

Jacard index:

index= intersection/union

y= [0 0 0 0 0 1 1 1 1 1]

y\_bar=[1 1 0 0 0 1 1 1 1 1]

index=8/10+10-8=0.66

So in jaccard if we go from 0 to 1 , accuracy of the model increases.

F1 score:

Confusion matrix:

true label churn1 6TP 9FN

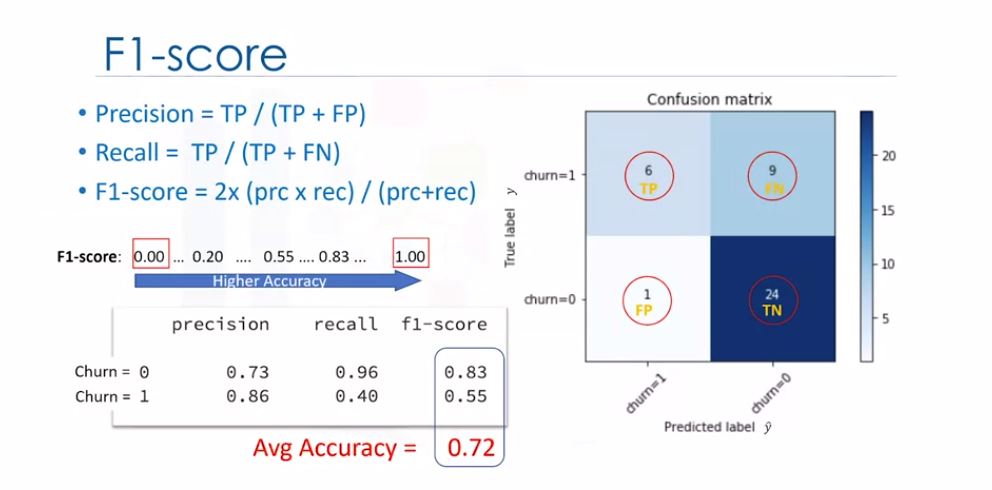
churn0 1FP 24TN

churn1 churn0

Predicted label

6 are true predicted values for churn =1

9 is the error because true was churn 1 but predicted churn 0.



Precision=TP/TP+FP

Measure of accuracy

Recall=TP/TP+FN

True postive rate

F1-score=2(prc\*recall)/prc+recall

So F1-score near to 1 is better.

So in F1 if we go from 0 to 1 , accuracy of the model increases.

Both Jaccard and F1 score are used for multi classifiers as well.

Log loss:

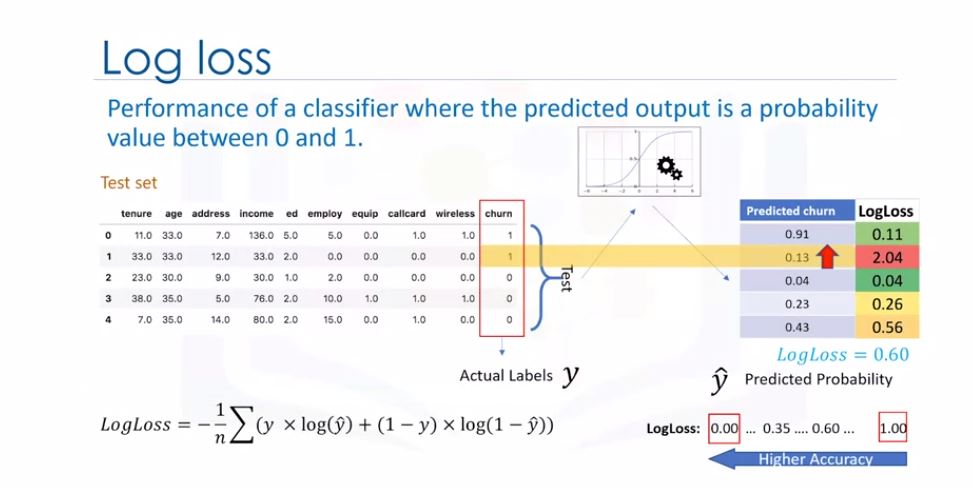
Sometimes output of the labels are not the value but the probability like in the

logistic regression.

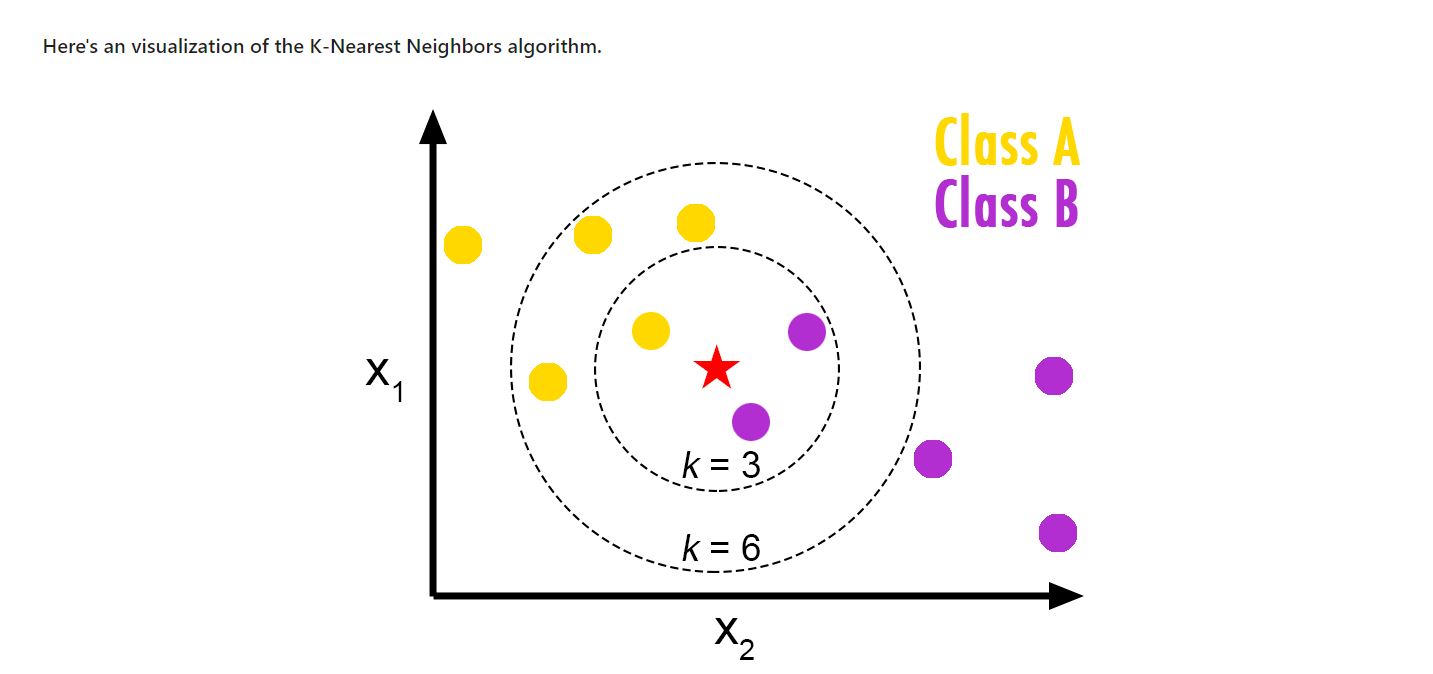
So use the log loss fromula to find all the distance and them take the average to check

the accuracy of the model.

So log loss near to 0 is better.



**Practice:**



**Code:**

//IMPORTING LIBRARIES

import itertools

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.ticker import NullFormatter

import pandas as pd

import numpy as np

import matplotlib.ticker as ticker

from sklearn import preprocessing

%matplotlib inline

Now about the data set , the telecommunication company wants to categorise the customers into 4 types.

The target field, called **custcat**, has four possible values that correspond to the four customer groups, as follows: 1- Basic Service 2- E-Service 3- Plus Service 4- Total Service

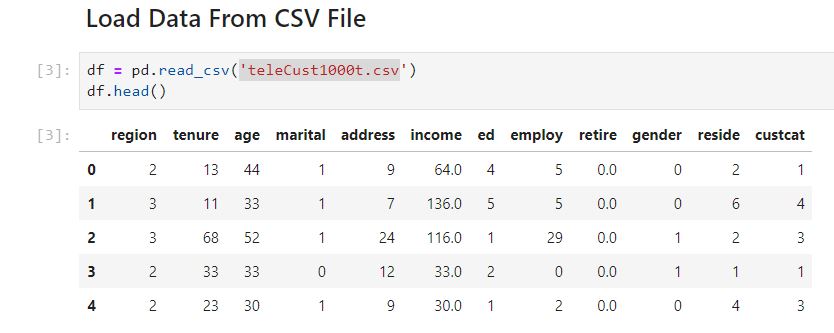
Download csv file from IBM storage.

!wget -O teleCust1000t.csv <https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/teleCust1000t.csv>

Load the csv FILE

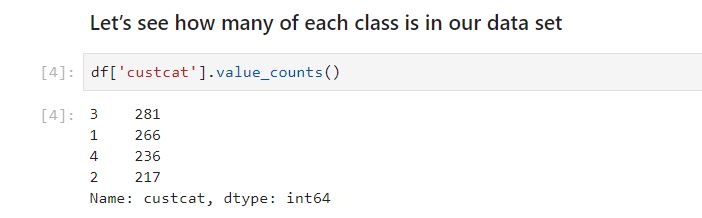
df = pd.read\_csv('teleCust1000t.csv’)

df.head()



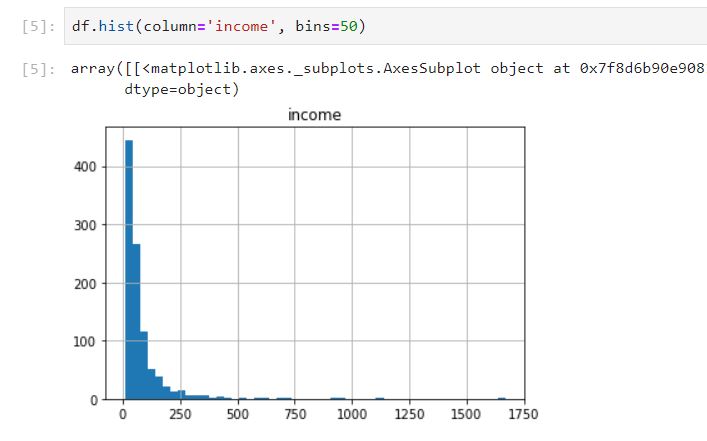
To get value categorised wise that how many customers are there in each category use this function

df[‘custcat’].value\_count()



Hist:

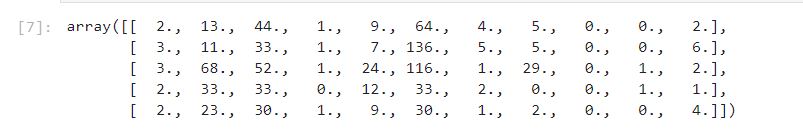
df.hist(column=‘income’,bins=50)



Define feature set

X=df[['region', 'tenure','age', 'marital', 'address', 'income', 'ed', 'employ','retire', 'gender', 'reside']].values

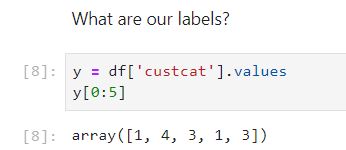
X[0:5] #give array from index 0 to 5 only



Labels

Y=df[[‘custcat’]].values

Y[0:5]

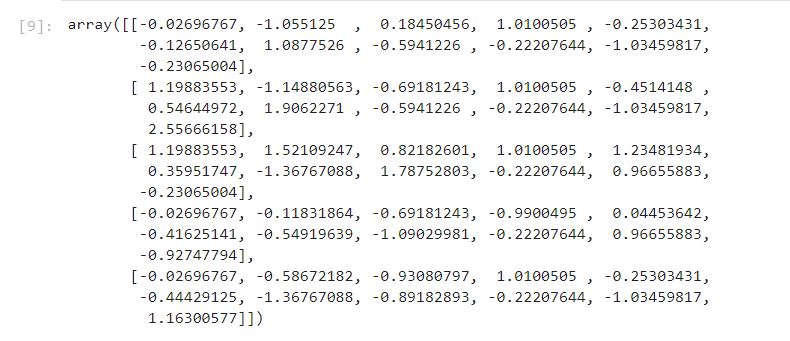


#Normalize data

To standardized the data with zero standard deviation and unit mean we use this.

X=preprocessing.StandardScaler().fit(X).transform(X.astype(float))

X[0:5]



**Train Test Split**

Out of Sample Accuracy is the percentage of correct predictions that the model makes on data that that the model has NOT been trained on. Doing a train and test on the same dataset will most likely have low out-of-sample accuracy, due to the likelihood of being over-fit.

It is important that our models have a high, out-of-sample accuracy, because the purpose of any model, of course, is to make correct predictions on unknown data. So how can we improve out-of-sample accuracy? One way is to use an evaluation approach called Train/Test Split. Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, you train with the training set and test with the testing set.

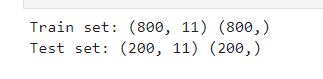
This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is not part of the dataset that have been used to train the data. It is more realistic for real world problems.

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,text\_size=0.2,random\_state=4)

print(‘X set size :’,X\_train.shape,X\_test.shape)

print(‘Y set size :’,Y\_train.shape,Y\_test.shape)



**Now main part of classification comes on**

#Importing libraries of KNN

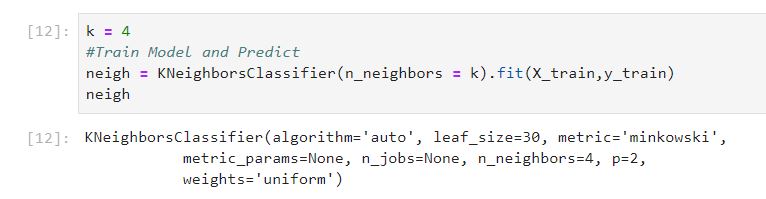
from sklearn.neighbors import KNeighborsClassifier

#Train model

k=4

neigh=KNeighborsClassifier(n\_neighbors = k).fit(X\_train,Y\_train)

neigh



#Now predicting the values

yhat=neigh.predict(X\_test)

yhat[0:5]



#accuracy score

from sklearn import metrics

print(‘Train accurracy’,metrics.accuraty\_score(y\_train,neigh.predict(X\_train)))

print(‘Test score’,metrics.accuracy\_score(y\_test,y\_hat))

**To find the best K**

first take k=1 find its accuracy then increment K and repeat the whole process.

Ks=10

mean\_acc=np.zeros(Ks-1)

std\_acc=np.zeros(Ks-1)

for n in range(1,Ks):

neigh = KNeighborsClassifier(n\_neighbors = n).fit(X\_train,y\_train)

yhat=neigh.predict(X\_test)

mean\_acc[n-1]=metrics.accuracy\_score(y\_test,y\_hat)

std\_acc[n-1]= np.std(yhat==y\_test)/np.sqrt(yhat.shape[0])

#Plot graph

plt.plot(range(1,Ks),mean\_acc,’g’)

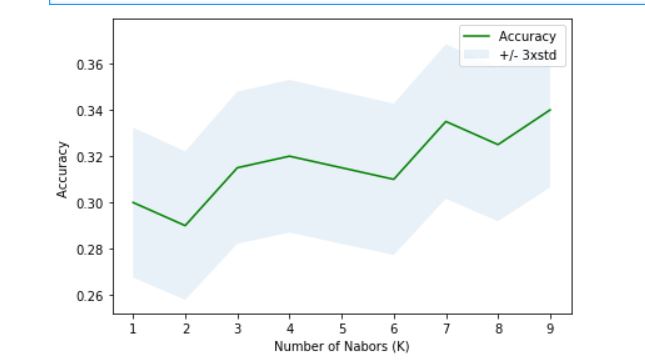
plt.fill\_between(range(1,Ks),mean\_acc - 1 \* std\_acc,mean\_acc + 1 \* std\_acc, alpha=0.10)

plt.legend(('Accuracy ', '+/- 3xstd'))

plt.xlabel(‘K’)

plt.ylabel(‘accuracy’)

plt.show()



print(‘The maximum accuracy at value of k=’,mean\_acc.argmax()+1,’ with accuracy score : ’,mean\_acc.max())