# XGBoost

#good for large dataset

#herefeature scaling not require but in deep learning its compulsary

#best model in term of good accuracy model and fast execution speed

#high performance, keep all interpration in model, high speed

#--------------------------------------

# Install xgboost following the instructions on this link: http://xgboost.readthedocs.io/en/latest/build.html#

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Churn\_Modelling.csv')

X = dataset.iloc[:, 3:13].values

y = dataset.iloc[:, 13].values

# Encoding categorical data

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder\_X\_1 = LabelEncoder()

X[:, 1] = labelencoder\_X\_1.fit\_transform(X[:, 1])

labelencoder\_X\_2 = LabelEncoder()

X[:, 2] = labelencoder\_X\_2.fit\_transform(X[:, 2])

onehotencoder = OneHotEncoder(categorical\_features = [1])

X = onehotencoder.fit\_transform(X).toarray()

X = X[:, 1:]

# Splitting the dataset into the Training set and Test set

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

# ---------Fitting XGBoost to the Training set

from xgboost import XGBClassifier

classifier = XGBClassifier() #you can do some parameter tunning as did grid search..contril+i(tunning on different parameter)

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

# Predicting the Test set results

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

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)#accuracy(1729/2000)

# Applying k-Fold Cross Validation

from sklearn.model\_selection import cross\_val\_score

accuracies = cross\_val\_score(estimator = classifier, X = X\_train, y = y\_train, cv = 10)

accuracies.mean()#accuracy is 86%

accuracies.std()# standard deviation is 1%