# Grid Search

#how do i now which parameter we should select during machine learning model

#how do i know which model choose for my machine learning problem.

#here aim is to improve model performance

#in ML,first tupe of parameter is parameter it learn during machine learning and second type of parameter is parameter we choose i.e hyperparameter ex kernal,svm model, penalty parameter and some regularization parameter.

#

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

#1)look at dependent variable, is its have continous outcome or categorical outcome///--

#it continous outcome then problem is regression problem

#if it is categorical outcome then it have classification problem.

#dont have dependent variable then it is clustering problem

#2) my problem is linear problem or non-linear problem,, for large amount of dataset its defficult to figureout data is linearly seperable or rather choose linear model like SVM if doing classification or non-linear model if doing kernal-SVM,, this question cab be answered by technique called grid search

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

# 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 = 0.25, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting Kernel SVM to the Training set

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random\_state = 0)

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)

# 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()

accuracies.std()

# Applying Grid Search to find the best model and the best parameters

from sklearn.model\_selection import GridSearchCV #cintrol+i after SVM function

parameters = [{'C': [1, 10, 100, 1000], 'kernel': ['linear']},#for linear model #in [] we put two dictionary, c -penalty parameter(more c prevent overfitting but not too less i.e will create underfitting)

{'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]}]# put non-linear model and several gaama value

grid\_search = GridSearchCV(estimator = classifier, #create grid object

param\_grid = parameters,

scoring = 'accuracy',

cv = 10, #take 10 accuracy from k-fold cross validation

n\_jobs = -1) #on large data set

grid\_search = grid\_search.fit(X\_train, y\_train) #fit to our training set

best\_accuracy = grid\_search.best\_score\_ #to get best accuracy score i.e 90% (mean of 10 fold)

best\_parameters = grid\_search.best\_params\_ #best list i.e optimal parameter find by grid search

#===============================================

##you can use Gris search classifier or take bestTune Hypperparameter from Gris search classifier and build your own classifier (SVM classifier)above

#==============================================================

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Kernel SVM (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Kernel SVM (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()