**PRACTICAL NO – 4**

**TITLE- SVM**

**THEORY**

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side*.*

There are kernels present in SVM:

we have different types of the kernels

1)linear--straight line

2)polynomial----polyline

3)radial basis function---rounded but separately rooted

**PROBLEM STATEMENT**- To develop an algorithm for SVM Classifier

**ALGORITHM-**

1. import libs
2. import dataset
3. Split
4. Normalization
5. fitting the SVM
6. predicting
7. confusion matrix
8. visualize

**CODE-**

# -\*- coding: utf-8 -\*-

"""

Created on Mon Feb 25 12:40:32 2019

@author: RoKu

"""

#import libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#import dataset

dataset = pd.read\_csv('ads\_v.csv')

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

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

#split the dataset

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)

#perform feature scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

#fit the SVM to the training set

from sklearn.svm import SVC

classifier = SVC(kernel='rbf',degree=3,random\_state=0)

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

#predicting the results

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

#deriving confusion matrix

from sklearn.metrics import confusion\_matrix

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

# 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('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('SVM (Testing set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**RESULTS**-

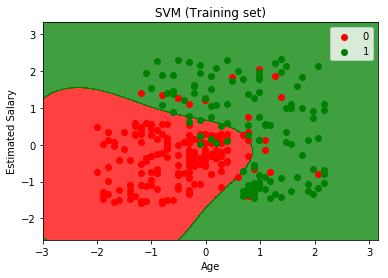
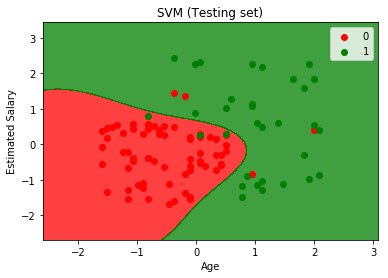
 

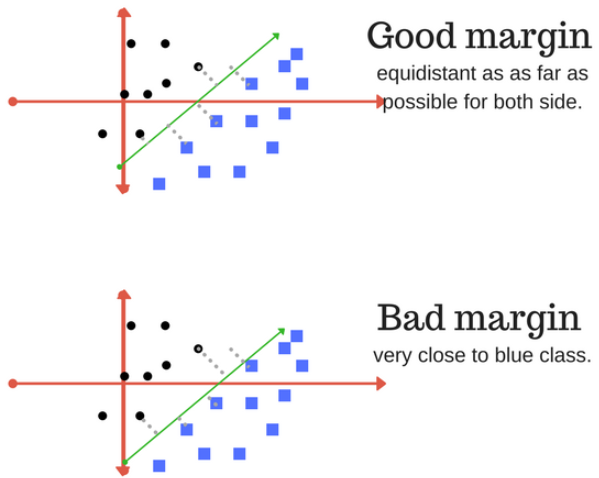
Fig.1- Visualization

**CONCLUSIONS-**

Important characteristic of SVM classifier. SVM to core tries to achieve a good margin.

***A margin is a separation of line to the closest class points.***

A ***good margin*** is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.



|  |  |
| --- | --- |
| **SUBMISSION DATE**-  /0/2019 | **SIGN OF COURSE INSTRUCTOR**- |
| **ROLL NO OF THE STUDENT-**  **I-62** | **NAME OF THE STUDENT-**  **Rohit Kulkarni** |