**SUPPORT VECTOR MACHINE (SVC)**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

x = df.iloc[:, :-1].values

y = df.iloc[:, -1].values

***splitting 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)

feature scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

print(x\_train)

print(x\_test)

***training the SVM model on the training set***

from sklearn.svm import SVC

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

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

***predicting a random value***

print(classifier.predict(sc.transform([[30, 87000]])))

predicting test set results

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

print(np.concatenate((y\_test.reshape(len(y\_test),1), y\_pred.reshape(len(y\_pred),1)),1))

***confusion matrix***

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score

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

print(cm)

accuracy\_score(y\_test, y\_pred)

precision\_score(y\_test, y\_pred)

recall\_score(y\_test, y\_pred)

f1\_score(y\_test, y\_pred)

***Visualizing training set results***

from matplotlib.colors import ListedColormap

x\_set, y\_set = sc.inverse\_transform(x\_train), y\_train

x1, x2 = np.meshgrid(np.arange(start = x\_set[:, 0].min() - 10, stop = x\_set[:, 0].max() + 10, step = 0.25),

np.arange(start = x\_set[:, 1].min() - 1000, stop = x\_set[:, 1].max() + 1000, step = 0.25))

plt.contourf(x1, x2, classifier.predict(sc.transform(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()

***Visualizing test set results***

from matplotlib.colors import ListedColormap

x\_set, y\_set = sc.inverse\_transform(x\_test), y\_test

x1, x2 = np.meshgrid(np.arange(start = x\_set[:, 0].min() - 10, stop = x\_set[:, 0].max() + 10, step = 0.25),

np.arange(start = x\_set[:, 1].min() - 1000, stop = x\_set[:, 1].max() + 1000, step = 0.25))

plt.contourf(x1, x2, classifier.predict(sc.transform(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 (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**SUPPORT VECTOR MACHINE (SVR)**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv('Position\_Salaries.csv')

df

x = df.iloc[:,1:-1].values

y = df.iloc[:, -1].values

print(x)

print(y)

y = y.reshape(len(y),1)

print(y)

***feature scaling***

from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

sc\_y = StandardScaler()

x = sc\_x.fit\_transform(x)

y = sc\_y.fit\_transform(y)

print(x)

print(y)

***training the SVR model***

from sklearn.svm import SVR

regressor = SVR(kernel = 'rbf')

regressor.fit(x,y)

***predicting a new result***

sc\_y.inverse\_transform(regressor.predict(sc\_x.transform([[6.5]])).reshape(-1,1))

***visualization***

plt.scatter(sc\_x.inverse\_transform(x), sc\_y.inverse\_transform(y), color='red')

plt.plot(sc\_x.inverse\_transform(x), sc\_y.inverse\_transform(regressor.predict(x).reshape(-1,1)), color='blue')

plt.title('Positions vs Salaries')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

***visualization (smooth curve)***

x\_grid = np.arange(min(sc\_x.inverse\_transform(x)), max(sc\_x.inverse\_transform(x)), 0.1)

x\_grid = x\_grid.reshape((len(x\_grid), 1))

plt.scatter(sc\_x.inverse\_transform(x), sc\_y.inverse\_transform(y), color = 'red')

plt.plot(x\_grid, sc\_y.inverse\_transform(regressor.predict(sc\_x.transform(x\_grid)).reshape(-1,1)), color = 'blue')

plt.title('Positions vs Salaries')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()