**SVM**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the datasets

df = pd.read\_csv('C:\\Users\\admin\\Desktop\\Social\_Network\_Ads.csv')

X = df.iloc[:, [2,3]]

Y = df.iloc[:, 4]

# 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)

#using linear kernel

from sklearn.svm import SVC

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

lin.fit(X\_train, Y\_train)

pred = lin.predict(X\_test)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(Y\_test,pred))

#using rbf kernel

from sklearn.svm import SVC

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

rbf.fit(X\_train, Y\_train)

pred = rbf.predict(X\_test)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(Y\_test,pred))

#using poly kernel

from sklearn.svm import SVC

poly = SVC(kernel = 'poly', degree=4)

poly.fit(X\_train, Y\_train)

pred = poly.predict(X\_test)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(Y\_test,pred))

plt.scatter(X\_test[:, 0], X\_test[:, 1],c=Y\_test)

# Create the hyperplane

w = lin.coef\_[0]

a = -w[0] / w[1]

xx = np.linspace(-2.5, 2.5)

yy = a \* xx - (lin.intercept\_[0]) / w[1]

# Plot the hyperplane

plt.plot(xx, yy)

plt.axis("off"), plt.show();

K-MEANS

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

mydata **=** pd**.**read\_csv("diabetes.csv")

*# mydata.head()*

x**=**mydata**.**iloc[:,[3,4]]**.**values

**from** sklearn.cluster **import** KMeans

km**=**KMeans(n\_clusters**=**4)

km**.**fit(x)

pred**=**km**.**predict(x)

pred

plt**.**scatter(x[pred**==**0,0],x[pred**==**0,1],c**=**'blue',label**=**'cluster 1')

plt**.**scatter(x[pred**==**1,0],x[pred**==**1,1],c**=**'red',label**=**'cluster 2')

plt**.**scatter(x[pred**==**2,0],x[pred**==**2,1],c**=**'green',label**=**'cluster 3')

plt**.**scatter(x[pred**==**3,0],x[pred**==**3,1],c**=**'yellow',label**=**'cluster 4')

plt**.**legend()

plt**.**show()

*#or method*

plt**.**scatter(x[:,0],x[:,1],c**=**pred)

plt**.**show()

LINEAR REGRESSION

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model

from sklearn.metrics import mean\_squared\_error, r2\_score

# Load the diabetes dataset

diabetes = datasets.load\_diabetes()

print(diabetes.feature\_names)

# Use only one feature (column) for simplicity

diabetes\_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets

diabetes\_X\_train = diabetes\_X[:-20]

diabetes\_X\_test = diabetes\_X[-20:]

# Split the targets into training/testing sets

diabetes\_y\_train = diabetes.target[:-20]

diabetes\_y\_test = diabetes.target[-20:]

# Create linear regression object

reg = linear\_model.LinearRegression()

# Train the model using the training sets

reg.fit(diabetes\_X\_train, diabetes\_y\_train)

# Make predictions using the testing set

diabetes\_y\_pred = reg.predict(diabetes\_X\_test)

# Print the coefficients and intercept

print("Coefficients: \n", reg.coef\_)

print("Intercept: \n", reg.intercept\_)

# Print mean squared error

print("Mean Squared Error: %.2f" % mean\_squared\_error(diabetes\_y\_test, diabetes\_y\_pred))

# Print coefficient of determination (r2 score)

print("Coefficient of determination: %.2f" % r2\_score(diabetes\_y\_test, diabetes\_y\_pred))

# Plot outputs

plt.scatter(diabetes\_X\_test, diabetes\_y\_test, color="black")

plt.plot(diabetes\_X\_test, diabetes\_y\_pred, color="blue", linewidth=3)

plt.xlabel("Age")

plt.ylabel("Diabetes Progression")

plt.xticks(())

plt.yticks(())

plt.show()