**Aim: Support Vector Machine (SVM)**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import statsmodels.api as sm**

**import statsmodels.formula.api as smf**

**from sklearn.linear\_model import LogisticRegression,LogisticRegressionCV**

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

**from sklearn.model\_selection import train\_test\_split,cross\_val\_score,cross\_val\_predict,ShuffleSplit,GridSearchCV**

**from sklearn.decomposition import PCA**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.preprocessing import scale**

**from sklearn import model\_selection**

**from sklearn.metrics import roc\_curve,auc,roc\_auc\_score**

**from sklearn import preprocessing**

**from sklearn.metrics import classification\_report**

**from sklearn.metrics import confusion\_matrix,accuracy\_score**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.ensemble import RandomForestClassifier,BaseEnsemble,GradientBoostingClassifier**

**from sklearn.svm import SVC,LinearSVC**

**import time**

**from matplotlib.colors import ListedColormap**

**from xgboost import XGBRegressor**

**from skompiler import skompile**

**from lightgbm import LGBMRegressor**

**pd.set\_option('display.max\_rows',1000)**

**pd.set\_option('display.max\_columns',1000)**

**pd.set\_option('display.width',1000)**

**df=pd.read\_csv("D:\AI practical\csv\diabetes.csv")**

**print(df.head(),"\n")**

**print(df.shape,"\n")**

**print(df.describe(),"\n\n")**

**X=df.drop('Outcome',axis=1)**

**y=df['Outcome']**

**X\_train=X.iloc[:600]**

**X\_test=X.iloc[600:]**

**y\_train=y[:600]**

**y\_test=y[600:]**

**print("X\_train Shape:",X\_train.shape)**

**print("X\_test Shape:",X\_test.shape)**

**print("y\_train Shape:",y\_train.shape)**

**print("y\_test Shape:",y\_test.shape)**

**support\_vector\_classifier=SVC(kernel="linear").fit(X\_train,y\_train)**

**print("\n\n",support\_vector\_classifier)**

**print("\n",support\_vector\_classifier.C)**

**print("\n",support\_vector\_classifier)**

**y\_pred=support\_vector\_classifier.predict(X\_test)**

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

**print("\n",cm,"\n")**

**print("Our Accuracy is: ",(cm[0][0]+cm[1][1])/(cm[0][0]+cm[1][1]+cm[0][1]+cm[1][0]),"\n")**

**print("Accuracy score: ",accuracy\_score(y\_test,y\_pred),"\n")**

**print("Classification Report : \n",classification\_report(y\_test,y\_pred),"\n")**

**#K-Fold Cross Validation**

**print(support\_vector\_classifier,"\n")**

**accuracies= cross\_val\_score(estimator=support\_vector\_classifier,X=X\_train,y=y\_train,cv=10)**

**print("Average Accuracy: {:.2f}%".format(accuracies.mean()\*100),"\n")**

**print("Standard Deviation of Accuracies: {:.2f}%".format(accuracies.std()\*100),"\n")**

**print(support\_vector\_classifier.predict(X\_test)[:10],"\n")**

**svm\_params = {"C":np.arange(1,20)}**

**svm= SVC(kernel="linear")**

**svm\_cv=GridSearchCV(svm,svm\_params,cv=8)**

**start\_time=time.time()**

**svm\_cv.fit(X\_train,y\_train)**

**elapsed\_time=time.time()-start\_time**

**print(f"Elapsed time for Support Vector Regression cross validation: "f"{elapsed\_time:.3f} seconds\n")**

**print("Best Score : ",svm\_cv.best\_score\_,"\n")**

**print("Best Parameter: ",svm\_cv.best\_params\_,"\n")**

**svm\_tuned=SVC(kernel="linear",C=2).fit(X\_train,y\_train)**

**print(svm\_tuned,"\n")**

**y\_pred=svm\_tuned.predict(X\_test)**

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

**print(cm,"\n")**

**print("Our Accuracy is: ",(cm[0][0]+cm[1][1])/(cm[0][0]+cm[1][1]+cm[0][1]+cm[1][0]),"\n")**

**print("Accuracy score: ",accuracy\_score(y\_test,y\_pred),"\n")**

**print("Classification Report : \n",classification\_report(y\_test,y\_pred),"\n")**

**Output:-**





