**Code 8 (SVM)**

from google.colab import drive

drive.mount('/content/drive')

import warnings

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

path="/content/drive/MyDrive/Machine Learning (ML)/pulsar\_stars.csv"

df=pd.read\_csv(path)

df.head()

df

df.isnull().sum()

df.shape

df.info()

df.describe()

df.columns

df.columns = df.columns.str.strip()

df.columns

df.columns = ['IP Mean', 'IP Sd', 'IP Kurtosis', 'IP Skewness', 'DM-SNR Mean', 'DM-SNR Sd', 'DM-SNR Kurtosis', 'DM-SNR Skewness', 'target\_class']

df.columns

df['target\_class'].value\_counts()

df['target\_class'].value\_counts()/np.float64(len(df))

round(df.describe(),2)

plt.figure(figsize=(24,20))

plt.subplot(4, 2, 1)

fig = df.boxplot(column='IP Mean')

fig.set\_title('')

fig.set\_ylabel('IP Mean')

plt.subplot(4, 2, 2)

fig = df.boxplot(column='IP Sd')

fig.set\_title('')

fig.set\_ylabel('IP Sd')

plt.subplot(4, 2, 3)

fig = df.boxplot(column='IP Kurtosis')

fig.set\_title('')

fig.set\_ylabel('IP Kurtosis')

plt.subplot(4, 2, 4)

fig = df.boxplot(column='IP Skewness')

fig.set\_title('')

fig.set\_ylabel('IP Skewness')

plt.subplot(4, 2, 5)

fig = df.boxplot(column='DM-SNR Mean')

fig.set\_title('')

fig.set\_ylabel('DM-SNR Mean')

plt.subplot(4, 2, 6)

fig = df.boxplot(column='DM-SNR Sd')

fig.set\_title('')

fig.set\_ylabel('DM-SNR Sd')

plt.subplot(4, 2, 7)

fig = df.boxplot(column='DM-SNR Kurtosis')

fig.set\_title('')

fig.set\_ylabel('DM-SNR Kurtosis')

plt.subplot(4, 2, 8)

fig = df.boxplot(column='DM-SNR Skewness')

fig.set\_title('')

fig.set\_ylabel('DM-SNR Skewness')

plt.figure(figsize=(24,20))

plt.subplot(4, 2, 1)

fig = df['IP Mean'].hist(bins=20)

fig.set\_xlabel('IP Mean')

fig.set\_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 2)

fig = df['IP Sd'].hist(bins=20)

fig.set\_xlabel('IP Sd')

fig.set\_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 3)

fig = df['IP Kurtosis'].hist(bins=20)

fig.set\_xlabel('IP Kurtosis')

fig.set\_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 4)

fig = df['IP Skewness'].hist(bins=20)

fig.set\_xlabel('IP Skewness')

fig.set\_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 5)

fig = df['DM-SNR Mean'].hist(bins=20)

fig.set\_xlabel('DM-SNR Mean')

fig.set\_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 6)

fig = df['DM-SNR Sd'].hist(bins=20)

fig.set\_xlabel('DM-SNR Sd')

fig.set\_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 7)

fig = df['DM-SNR Kurtosis'].hist(bins=20)

fig.set\_xlabel('DM-SNR Kurtosis')

fig.set\_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 8)

fig = df['DM-SNR Skewness'].hist(bins=20)

fig.set\_xlabel('DM-SNR Skewness')

fig.set\_ylabel('Number of pulsar stars')

X = df.drop(['target\_class'], axis=1)

y = df['target\_class']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

X\_train.shape, X\_test.shape

cols = X\_train.columns

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

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

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

X\_train = pd.DataFrame(X\_train, columns=[cols])

X\_test = pd.DataFrame(X\_test, columns=[cols])

X\_train.describe()

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

svc = SVC()

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

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

print('Model accuracy score with default hyperparameters: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

svc = SVC(C=10000.0)

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

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

print('Model accuracy score with rbf kernal and C=1000.0: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

svc = SVC(C=100.0)

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

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

print('Model accuracy score with rbf kernal and C=100.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

**Run SVM with linear Kernal**

linear\_svc=SVC(kernel='linear', C=1.0)

linear\_svc.fit(X\_train,y\_train)

y\_pred\_test=linear\_svc.predict(X\_test)

print('Model accuracy score with linear kernal and C=1.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

linear\_svc=SVC(kernel='linear', C=100.0)

linear\_svc.fit(X\_train,y\_train)

y\_pred\_test=linear\_svc.predict(X\_test)

print('Model accuracy score with linear kernal and C=100.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

linear\_svc=SVC(kernel='linear', C=1000.0)

linear\_svc.fit(X\_train,y\_train)

y\_pred\_test=linear\_svc.predict(X\_test)

print('Model accuracy score with linear kernal and C=1.0000 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

print ('Training set score: {:.4f}'.format(linear\_svc.score(X\_train, y\_train)))

print ('Test set score: {:.4f}'.format(linear\_svc.score(X\_test, y\_test)))

**RUN SVM WITH PLOYNOMIAL KERNAL**

poly\_svc=SVC(kernel='poly', C=1.0)

poly\_svc.fit(X\_train,y\_train)

y\_pred=poly\_svc.predict(X\_test)

print('Model accuracy score with polynomial kernal and C=1.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

poly\_svc=SVC(kernel='poly', C=100.0)

poly\_svc.fit(X\_train,y\_train)

y\_pred=poly\_svc.predict(X\_test)

print('Model accuracy score with polynomial kernal and C=100.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

poly\_svc=SVC(kernel='poly', C=1000.0)

poly\_svc.fit(X\_train,y\_train)

y\_pred=poly\_svc.predict(X\_test)

print('Model accuracy score with polynomial kernal and C=1000.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

**RUN SVM WITH SIGMIOD KERNAL**

sigmoid\_svc=SVC(kernel='sigmoid',C=1.0)

sigmoid\_svc.fit(X\_train,y\_train)

y\_pred=sigmoid\_svc.predict(X\_test)

print('Model accuracy score with sigmoid kernal and C=1.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

sigmoid\_svc=SVC(kernel='sigmoid',C=100.0)

sigmoid\_svc.fit(X\_train,y\_train)

y\_pred=sigmoid\_svc.predict(X\_test)

print('Model accuracy score with Sigmoid kernal and C=100.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

sigmoid\_svc=SVC(kernel='sigmoid',C=1000.0)

sigmoid\_svc.fit(X\_train,y\_train)

y\_pred=sigmoid\_svc.predict(X\_test)

print('Model accuracy score with sigmoid kernal and C=1000.0 : {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))

from sklearn.metrics import confusion\_matrix

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

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Neagatives(FN) = ', cm[1,0])

import seaborn as sns

cm\_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],

                                 index=['Predict Positive:1', 'Predict Negative:0'])

sns.heatmap(cm\_matrix, annot=True, fmt='d', cmap='YlGnBu')

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred\_test))

TP = cm[0,0]

TN = cm[1,1]

FP = cm[0,1]

FN = cm[1,0]

classification\_accuracy = (TP + TN) / float(TP + TN + FP + FN)

print('Classification accuracy : {0:0.4f}'.format(classification\_accuracy))

classification\_error = (FP + FN) / float(TP + TN + FP + FN)

print('Classification error : {0:0.4f}'.format(classification\_error))