EXPERIMENT 8

Aim

To implement and evaluate a dataset using SVM based classification algorithm

Software Used

Google Colab

Program Code and Output

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline
```

```
[ ] data = '/content/pulsar_data_train.csv'
    #data = 'kaggle datasets download -d colearninglounge/predicting-pulsar-starintermediate'

    df = pd.read_csv(data)
```

```
[ ] # view dimensions of dataset

    df.shape
```

```
(12528, 9)
```

```
[ ] # let's preview the dataset

    df.head()
```

	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
0	121.156250	48.372971	0.375485	-0.013165	3.168896	18.399367	7.449874	65.159298	0.0
1	76.968750	36.175557	0.712898	3.388719	2.399666	17.570997	9.414652	102.722975	0.0
2	130.585938	53.229534	0.133408	-0.297242	2.743311	22.362553	8.508364	74.031324	0.0
3	156.398438	48.865942	-0.215989	-0.171294	17.471572	NaN	2.958066	7.197842	0.0
4	84.804688	36.117659	0.825013	3.274125	2.790134	20.618009	8.405008	76.291128	0.0

```
[ ] # view the column names of the dataframe

    col_names = df.columns

    col_names
```

```
Index([' Mean of the integrated profile',
       ' Standard deviation of the integrated profile',
       ' Excess kurtosis of the integrated profile',
       ' Skewness of the integrated profile', ' Mean of the DM-SNR curve',
       ' Standard deviation of the DM-SNR curve',
       ' Excess kurtosis of the DM-SNR curve', ' Skewness of the DM-SNR curve',
       'target_class'],
      dtype='object')
```

```
# check for missing values in variables

df.isnull().sum()
```

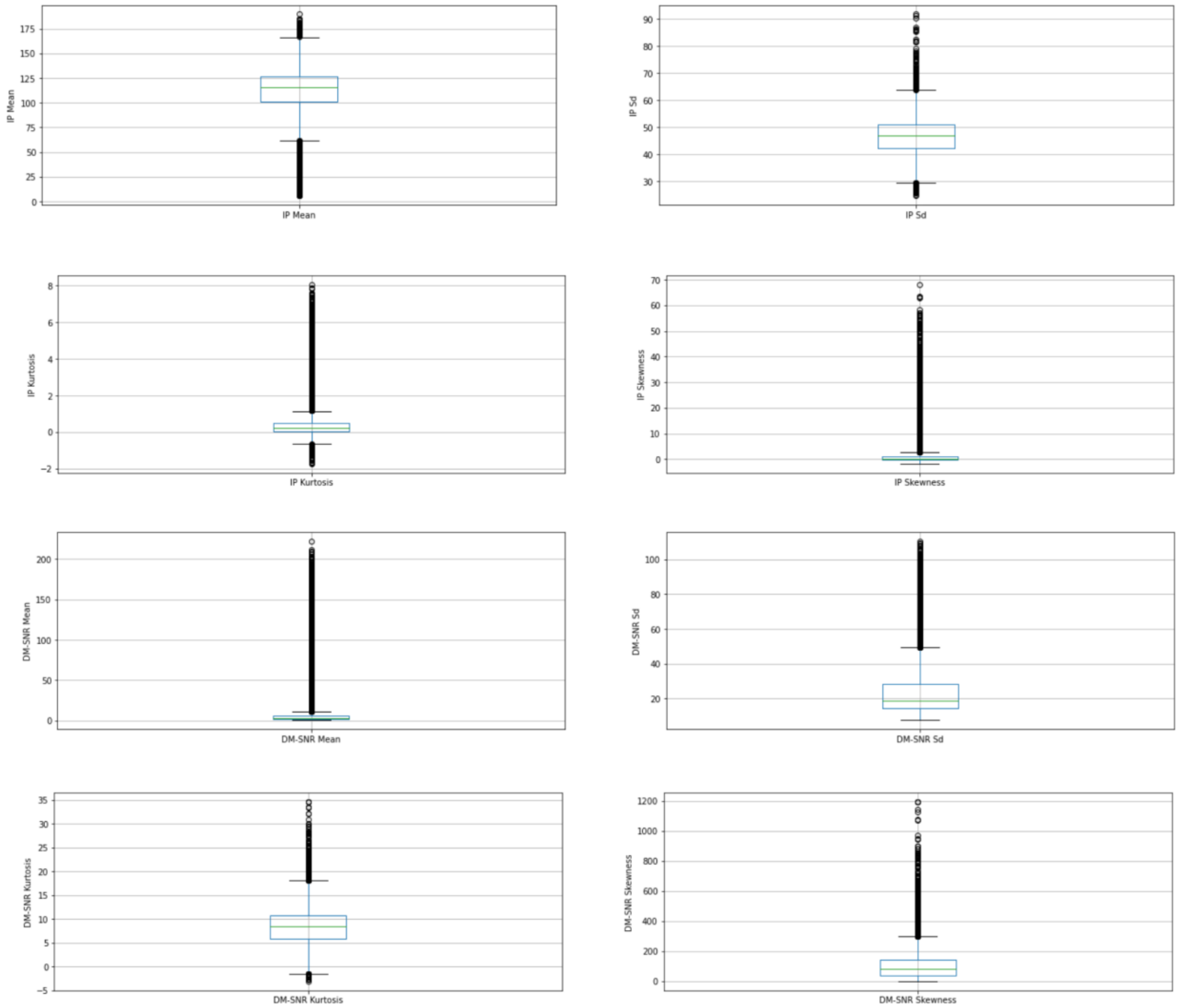
```
IP Mean                 0
IP Sd                   0
IP Kurtosis          1735
IP Skewness             0
DM-SNR Mean             0
DM-SNR Sd            1178
DM-SNR Kurtosis         0
DM-SNR Skewness       625
target_class            0
dtype: int64
```

```
# draw boxplots to visualize outliers

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')
```

```python
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')
```

```
Text(0, 0.5, 'DM-SNR Skewness')
```




```python
[ ]  X = df.drop(['target_class'], axis=1)

     y = df['target_class']
```

```
[ ] # split X and y into training and testing sets

    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)
```

```
[ ] # check the shape of X_train and X_test

    X_train.shape, X_test.shape
```

```
((10022, 8), (2506, 8))
```

```
[ ] 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()
```

	IP Mean	IP Sd	IP Kurtosis	IP Skewness	DM-SNR Mean	DM-SNR Sd	DM-SNR Kurtosis	DM-SNR Skewness
count	1.002200e+04	1.002200e+04	8.616000e+03	1.002200e+04	1.002200e+04	9.090000e+03	1.002200e+04	9.500000e+03
mean	-2.580698e-16	-7.770453e-16	-5.154239e-17	-1.595212e-17	-3.030902e-17	-9.028348e-17	-1.318708e-16	6.731457e-18
std	1.000050e+00	1.000050e+00	1.000058e+00	1.000050e+00	1.000050e+00	1.000055e+00	1.000050e+00	1.000053e+00
min	-4.059253e+00	-3.121855e+00	-2.060343e+00	-5.703669e-01	-4.225211e-01	-9.665725e-01	-2.526379e+00	-9.997646e-01
25%	-3.943394e-01	-6.101706e-01	-4.266826e-01	-3.175801e-01	-3.653436e-01	-6.094788e-01	-5.589324e-01	-6.565951e-01
50%	1.619199e-01	5.986146e-02	-2.415742e-01	-2.549120e-01	-3.355278e-01	-4.066791e-01	2.442155e-02	-2.086985e-01
75%	6.265131e-01	6.579129e-01	-1.143402e-02	-1.397269e-01	-2.459675e-01	1.018419e-01	5.276848e-01	3.194451e-01
max	3.045294e+00	6.647182e+00	7.026077e+00	1.045442e+01	7.074053e+00	4.281896e+00	5.769814e+00	1.009101e+01

```
[29] X_train = X_train.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
     #y_train = y_train.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
     X_test = X_test.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
```

```python
from sklearn.svm import SVC


# import metrics to compute accuracy
from sklearn.metrics import accuracy_score


# instantiate classifier with default hyperparameters
svc=SVC()


# fit classifier to training set
svc.fit(X_train,y_train)


# make predictions on test set
y_pred=svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with default hyperparameters: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with default hyperparameters: 0.9796
```

[ ]

```python
# instantiate classifier with rbf kernel and C=100
svc=SVC(C=100.0)


# fit classifier to training set
svc.fit(X_train,y_train)


# make predictions on test set
y_pred=svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with rbf kernel and C=100.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with rbf kernel and C=100.0 : 0.9804
```

[ ]

```python
# instantiate classifier with rbf kernel and C=1000
svc=SVC(C=1000.0)


# fit classifier to training set
svc.fit(X_train,y_train)


# make predictions on test set
y_pred=svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with rbf kernel and C=1000.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with rbf kernel and C=1000.0 : 0.9808
```

```python
print('Training set score: {:.4f}'.format(svc.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(svc.score(X_test, y_test)))
```

```
Training set score: 0.9871
Test set score: 0.9808
```

Discussion and Conclusion

Support Vector Machine classifier is implemented on a pulsar dataset. Dataset is scaled using z score normalisation. Dataset is visualised with Boxplots and checked for outliers, since the dataset contains outliers, so the value of C should be high while training the model. Soft-margin variant of SVM is used and in this case, we can have a few points incorrectly classified or classified with a margin less than 1. But for every such point, we have to pay a penalty in the form of C parameter, which controls the outliers. Low C implies we are allowing more outliers and high C implies less outliers. Since accuracy of train and test data is comparable so the problem of overfitting will not arise.

CRITERIA	TOTAL MARKS	MARKS OBTAINED	COMMENTS
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		