```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
data = pd.read_csv('pulsar_stars.csv')
```

```
data.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	Skew of DM c
0	140.562500	55.683782	-0.234571	-0.699648	3.199833	19.110426	7.975532	74.24
1	102.507812	58.882430	0.465318	-0.515088	1.677258	14.860146	10.576487	127.39
2	103.015625	39.341649	0.323328	1.051164	3.121237	21.744669	7.735822	63.17
3	136.750000	57.178449	-0.068415	-0.636238	3.642977	20.959280	6.896499	53.59

```
data.shape
```

```
(17898, 9)
```

```
col_name = data.columns
print(col_name)
```

```
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')
```

```
data.columns = data.columns.str.strip()
```

```
data.columns
```

```
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')
```

```
data.columns = ['IP Mean', 'IP Sd', 'IP Kurtosis', 'IP Skewness',
              'DM-SNR Mean', 'DM-SNR Sd', 'DM-SNR Kurtosis', 'DM-SNR Skewness', 'target_class']
```

```
data.columns
```

```
Index(['IP Mean', 'IP Sd', 'IP Kurtosis', 'IP Skewness', 'DM-SNR Mean',
       'DM-SNR Sd', 'DM-SNR Kurtosis', 'DM-SNR Skewness', 'target_class'],
      dtype='object')
```

Exploring the target Class

```
data['target_class'].value_counts()
```

```
0    16259
1     1639
Name: target_class, dtype: int64
```

```
data['target_class'].value_counts()/np.float(len(data))*100
```

```
<ipython-input-70-72c7094f3a90>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  data['target_class'].value_counts()/np.float(len(data))*100
0    90.842552
1     9.157448
Name: target_class, dtype: float64
```

Summary of dataset

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17898 entries, 0 to 17897
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   IP Mean          17898 non-null  float64
 1   IP Sd            17898 non-null  float64
 2   IP Kurtosis      17898 non-null  float64
 3   IP Skewness      17898 non-null  float64
 4   DM-SNR Mean      17898 non-null  float64
 5   DM-SNR Sd        17898 non-null  float64
 6   DM-SNR Kurtosis  17898 non-null  float64
 7   DM-SNR Skewness  17898 non-null  float64
 8   target_class     17898 non-null  int64  
dtypes: float64(8), int64(1)
memory usage: 1.2 MB
```

Exploring the missing values

```
data.isnull().sum()
```

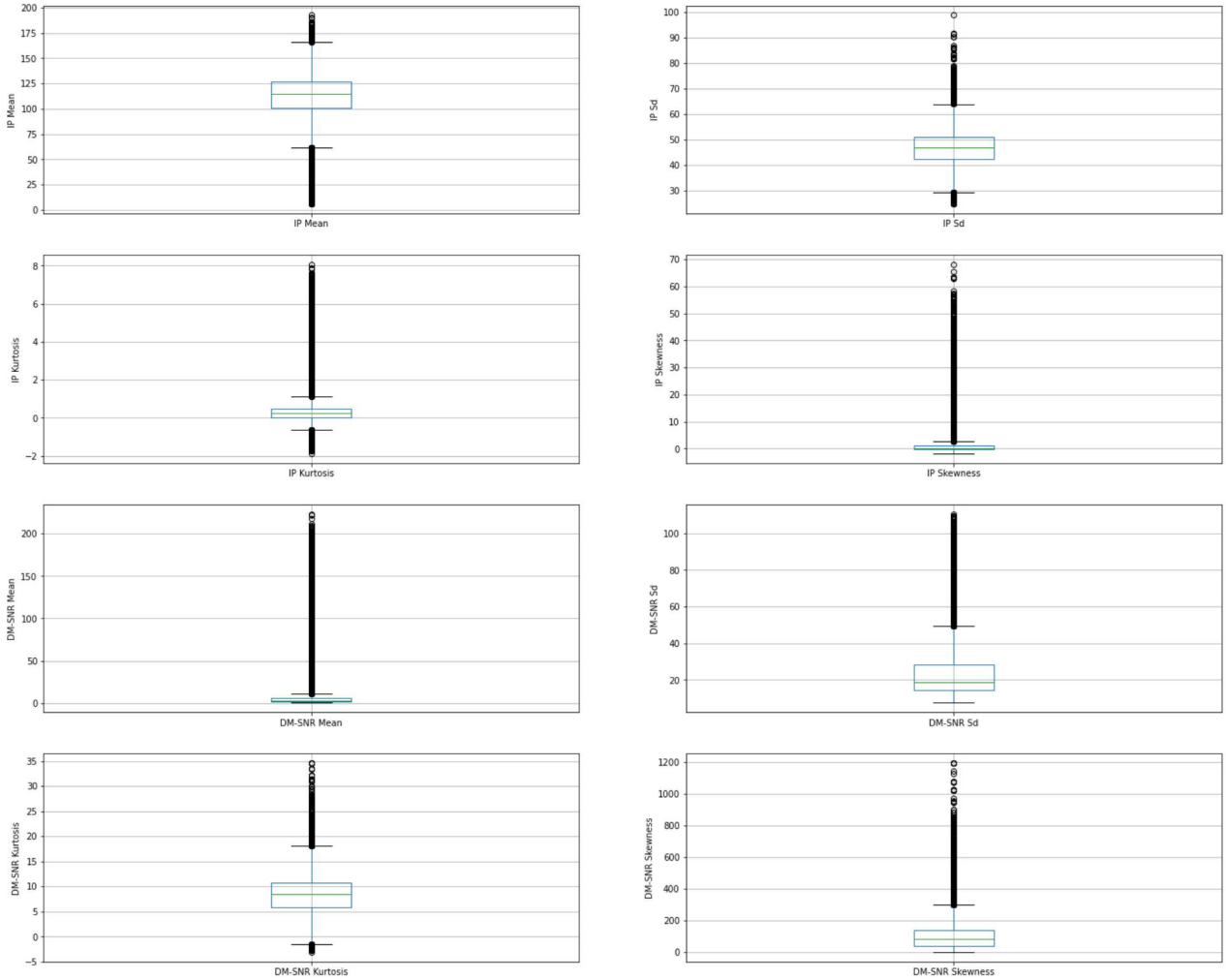
```
IP Mean            0
IP Sd              0
IP Kurtosis        0
IP Skewness        0
DM-SNR Mean        0
DM-SNR Sd          0
DM-SNR Kurtosis    0
DM-SNR Skewness    0
target_class       0
dtype: int64
```

Data Viaulaization

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

plt.subplot(4,2,1)
fig = data.boxplot(column='IP Mean')
fig.set_title('')
fig.set_ylabel('IP Mean')

plt.subplot(4,2,2)
fig = data.boxplot(column='IP Sd')
fig.set_title('')
fig.set_ylabel('IP Sd')

plt.subplot(4,2,3)
fig = data.boxplot(column='IP Kurtosis')
fig.set_title('')
fig.set_ylabel('IP Kurtosis')

plt.subplot(4,2,4)
fig = data.boxplot(column='IP Skewness')
```

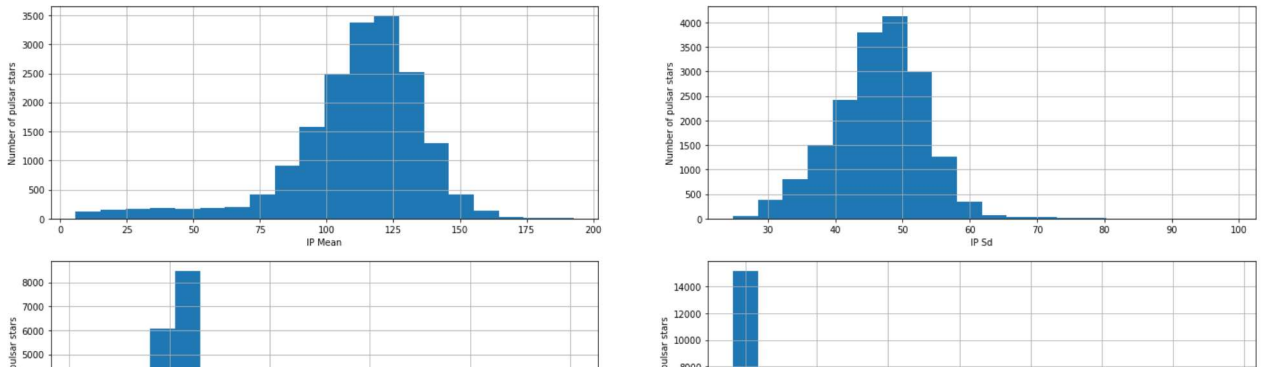
```
fig.set_title('')
fig.set_ylabel('IP Skewness')

plt.subplot(4,2,5)
fig = data.boxplot(column='DM-SNR Mean')
fig.set_title('')
fig.set_ylabel('DM-SNR Mean')

plt.subplot(4,2,6)
fig = data.boxplot(column='DM-SNR Sd')
fig.set_title('')
fig.set_ylabel('DM-SNR Sd')

plt.subplot(4,2,7)
fig = data.boxplot(column='DM-SNR Kurtosis')
fig.set_title('')
fig.set_ylabel('DM-SNR Kurtosis')

plt.subplot(4,2,8)
fig = data.boxplot(column='DM-SNR Skewness')
fig.set_title('')
fig.set_ylabel('DM-SNR Skewness')
```

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



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

plt.subplot(4, 2, 1)
fig = data['IP Mean'].hist(bins=20)
fig.set_xlabel('IP Mean')
fig.set_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 2)
fig = data['IP Sd'].hist(bins=20)
fig.set_xlabel('IP Sd')
fig.set_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 3)
fig = data['IP Kurtosis'].hist(bins=20)
fig.set_xlabel('IP Kurtosis')
fig.set_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 4)
fig = data['IP Skewness'].hist(bins=20)
fig.set_xlabel('IP Skewness')
fig.set_ylabel('Number of pulsar stars')

plt.subplot(4, 2, 5)
fig = data['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 = data['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 = data['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 = data['DM-SNR Skewness'].hist(bins=20)
fig.set_xlabel('DM-SNR Skewness')
fig.set_ylabel('Number of pulsar stars')
```

```
Text(0, 0.5, 'Number of pulsar stars')
```



Train Test Split

```
X = data.drop(['target_class'], axis=1)
y = data['target_class']
```

```
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
```

```
(3580, 8)
```

Model Training

```
svc = SVC()

svc.fit(X_train, y_train)
```

```
SVC()
```

Accuracy of the Model

```
# Training Accuracy
y_pred_train=svc.predict(X_train)
print("The accuracy of model in training data: ", accuracy_score(y_train, y_pred_train))
# Testing Accuracy
y_pred=svc.predict(X_test)
print("The accuracy of model with default hyperparameters:", accuracy_score(y_test, y_pred))
```

```
The accuracy of model in training data:  0.9713647157424221
The accuracy of model with default hyperparameters: 0.9784916201117319
```