[CM5]

Data Cleaning

Checking for null / NaN values (missing data)

[14]:
```
# checking for any null / NaN values
df_iris.isnull().values.any()
```

[14]:
```
True
```

[15]:
```
# checking for any null / NaN values
df_iris.isna().sum()
```

[15]:
```
sepal_length     0
sepal_width      7
petal_length    11
petal_width      1
species          0
dtype: int64
```

We notice few null / NaN values in the dataset. We can handle these by replacing with feature mean

Checking for negative values

We notice few negative values in the petal width feature of the dataset (from initial data exploration) . Ideally length and width are expected to be positive. Hence, it's better handling the anamolous negative values by replacing with feature mean.

[16]:
```
# checking for negative values
df_iris['petal_width'].sort_values()
```

[16]:
```
67   -0.072203
6    -0.042428
34    0.020731
43    0.091698
80    0.104012
        …
71    2.424502
62    2.478509
44    2.554837
47    2.603123
79         NaN
Name: petal_width, Length: 105, dtype: float64
```

[17]:
```
# replacing negative values with NaN (Will be later replaced with feature mean)
for index in df_iris[df_iris['petal_width'] < 0].index:
    df_iris.loc[index, 'petal_width'] = np.nan
```

[18]:
```
df_iris.describe()
```

[18]:

	sepal_length	sepal_width	petal_length	petal_width
count	105.000000	98.000000	94.000000	102.000000
mean	5.858909	3.052443	3.880252	1.230288

```
std         0.861638     0.441646      1.778404     0.776025
min         4.344007     1.946010      1.253850     0.020731
25%         5.159145     2.769449      1.549032     0.340189
50%         5.736104     3.045434      4.349280     1.359332
75%         6.435413     3.238732      5.097752     1.832747
max         7.795561     4.409565      6.768611     2.603123
```

[19]:
```
# replacing all NaN values with feature mean
for column in df_iris.columns[1:-1]:
    df_iris[column].fillna(value=df_iris[column].mean(), inplace=True)
```

[20]:
```
# check if there are any null / NaN values
df_iris.isnull().values.any()
```

[20]:
```
False
```

[21]:
```
# check if there are any null / NaN values
df_iris.isna().sum()
```

[21]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

[22]:
```
df_iris.describe()
```

[22]:
```
       sepal_length  sepal_width  petal_length  petal_width
count    105.000000   105.000000    105.000000   105.000000
mean       5.858909     3.052443      3.880252     1.230288
std        0.861638     0.426524      1.681726     0.764751
min        4.344007     1.946010      1.253850     0.020731
25%        5.159145     2.794790      1.592887     0.343669
50%        5.736104     3.052443      4.089166     1.331797
75%        6.435413     3.234061      5.062244     1.817211
max        7.795561     4.409565      6.768611     2.603123
```

Data Cleaning :

- the NaN values (missing values) were replaced with feature mean.
- the negative values (in case of petal width) were replaced with feature mean.

If we attempt to drop the missing and negative values, the performance of the classifier was observed to be low. Moreover, dropping the values reduces the size of the dataset affecting performance.