Abdul Saboor (20L-1113 | BDS-5A1)

Pre-requisites

```
#importing ibraries to use
import pandas as pd
import numpy as np
```

```
data=pd.read_excel('my_iris.xlsx')
```

```
# checking data
data.describe()
```

	Unnamed: 0	sepal depth	sepal diameter	petal depth	petal diameter	sepa
count	150.000000	130.000000	130.000000	130.000000	100.000000	1!
mean	74.500000	5.901538	3.005385	4.115385	1.453000	
std	43.445368	0.835082	0.410013	1.621200	0.685117	
min	0.000000	4.300000	2.000000	1.100000	0.100000	
25%	37.250000	5.125000	2.800000	3.350000	1.100000	
50%	74.500000	5.850000	3.000000	4.500000	1.500000	
75%	111.750000	6.475000	3.200000	5.200000	2.000000	
max	149.000000	7.900000	4.100000	6.900000	2.500000	

Data Cleaning

Dropping redundant columns

```
data=data.drop([data.columns[0]],axis=1)
data.describe()

# Till now only first column is dropped as indexes of dataframe are duplicating with i
#  So dropping index column will have no effect on data.
```

	sepal depth	sepal diameter	petal depth	petal diameter	sepal length	sep
count	130.000000	130.000000	130.000000	100.000000	150.000000	
mean	5.901538	3.005385	4.115385	1.453000	11.686667	
std	0.835082	0.410013	1.621200	0.685117	1.656132	
min	4.300000	2.000000	1.100000	0.100000	8.600000	
25%	5.125000	2.800000	3.350000	1.100000	10.200000	
50%	5.850000	3.000000	4.500000	1.500000	11.600000	
75%	6.475000	3.200000	5.200000	2.000000	12.800000	

Structuring and Standardising data

```
data['sepal width']=data['sepal width'].apply(lambda x: -1*x) # for sepal width
data['petal width']=data['petal width'].apply(lambda x: -1*x) # for petal width
data.describe()

# The column of sepal width and petal width have all negative values, however a measu
# This may be considered a human error and all values should be converted to positive
```

	sepal depth	sepal diameter	petal depth	petal diameter	sepal length	sep
count	130.000000	130.000000	130.000000	100.000000	150.000000	
mean	5.901538	3.005385	4.115385	1.453000	11.686667	
std	0.835082	0.410013	1.621200	0.685117	1.656132	
min	4.300000	2.000000	1.100000	0.100000	8.600000	
25%	5.125000	2.800000	3.350000	1.100000	10.200000	
50%	5.850000	3.000000	4.500000	1.500000	11.600000	
75%	6.475000	3.200000	5.200000	2.000000	12.800000	
max	7.900000	4.100000	6.900000	2.500000	15.800000	

Filling missing values

```
# for petal depth column
temp=pd.DataFrame({'original':data['petal depth'],'interpolation':data['petal depth'].
temp.describe()
```

	original	interpolation	mean	median
count	130.000000	149.000000	150.000000	150.000000
mean	4.115385	3.780201	4.115385	4.166667
std	1.621200	1.751021	1.508475	1.514169
min	1.100000	1.100000	1.100000	1.100000
25%	3.350000	1.600000	3.825000	3.825000
50%	4.500000	4.400000	4.350000	4.500000
75%	5.200000	5.100000	5.100000	5.100000

```
# for petal diameter column
temp=pd.DataFrame({'original':data['petal diameter'],'interpolation':data['petal diame
temp.describe()
```

	original	interpolation	mean	median
count	100.000000	149.000000	150.000000	150.000000
mean	1.453000	1.207383	1.453000	1.468667
std	0.685117	0.747231	0.558456	0.558898
min	0.100000	0.100000	0.100000	0.100000
25%	1.100000	0.350000	1.300000	1.300000
50%	1.500000	1.300000	1.453000	1.500000
75%	2.000000	1.800000	1.800000	1.800000
max	2.500000	2.500000	2.500000	2.500000

```
# for sepal depth column
temp=pd.DataFrame({'original':data['sepal depth'],'interpolation':data['sepal depth'].
temp.describe()
```

	original	interpolation	mean	median

```
# for sepal diameter column
temp=pd.DataFrame({'original':data['sepal diameter'],'interpolation':data['sepal diame
temp.describe()
```

	original	interpolation	mean	median
count	130.000000	149.000000	150.000000	150.000000
mean	3.005385	3.046644	3.005385	3.004667
std	0.410013	0.413606	0.381504	0.381508
min	2.000000	2.000000	2.000000	2.000000
25%	2.800000	2.800000	2.800000	2.800000
50%	3.000000	3.000000	3.000000	3.000000
75%	3.200000	3.300000	3.200000	3.200000
max	4.100000	4.100000	4.100000	4.100000

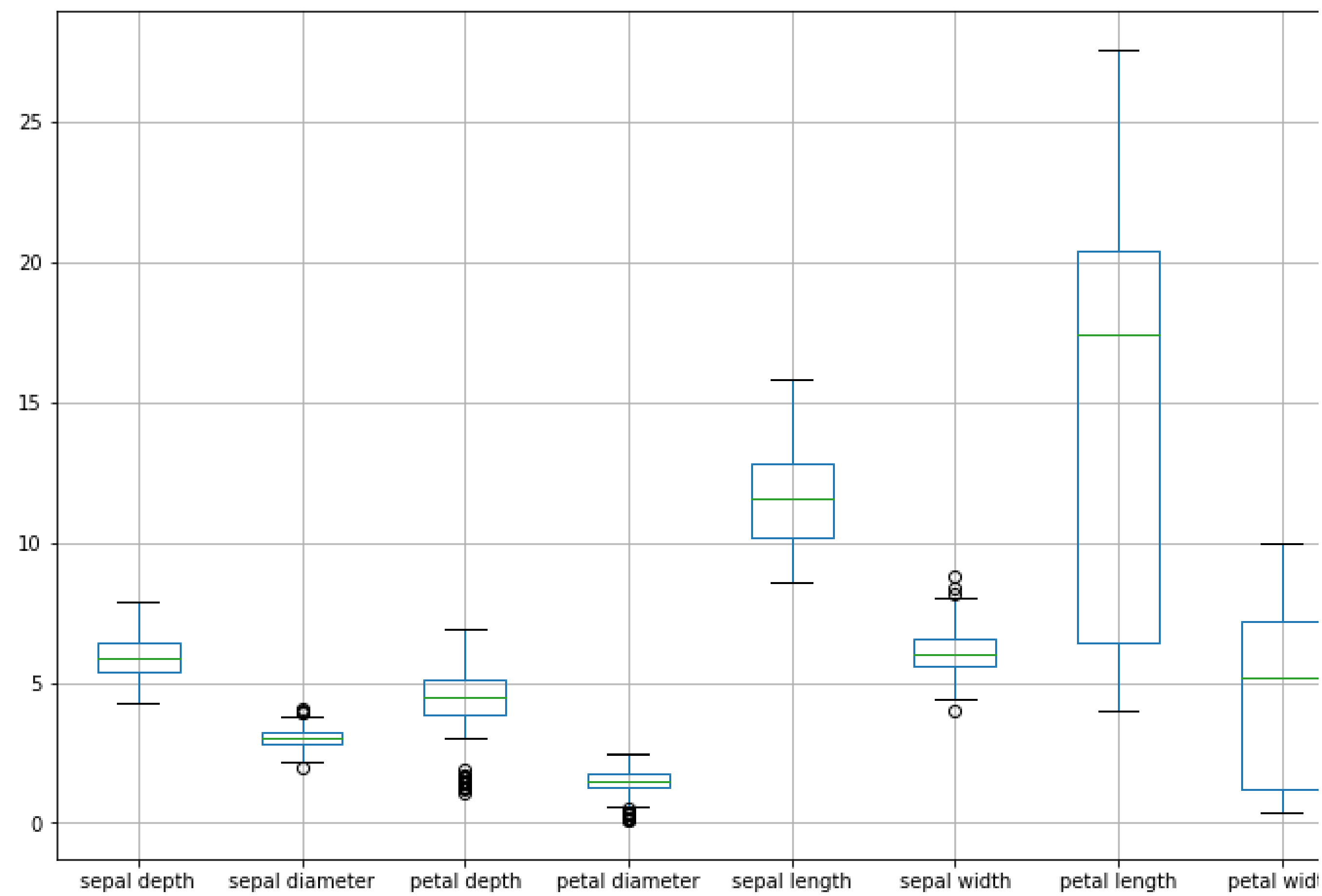
```
data['petal depth']=data['petal depth'].fillna(data['petal depth'].median())
data['petal diameter']=data['petal diameter'].fillna(data['petal diameter'].median())
data['sepal depth']=data['sepal depth'].fillna(data['sepal depth'].median())
data['sepal diameter']=data['sepal diameter'].fillna(data['sepal diameter'].median())

# Here replacing missing values with median over mean or interpolation is preferred a1
# of all three approaches. It can be observed that replacing with median values create
# in data as both mean and median deviate by minimum in this case.
data.describe()
```

	sepal depth	sepal diameter	petal depth	petal diameter	sepal length	sep
count	150.000000	150.000000	150.000000	150.000000	150.000000	
mean	5.894667	3.004667	4.166667	1.468667	11.686667	
std	0.777216	0.381508	1.514169	0.558898	1.656132	
min	4.300000	2.000000	1.100000	0.100000	8.600000	
25%	5.400000	2.800000	3.825000	1.300000	10.200000	
50%	5.850000	3.000000	4.500000	1.500000	11.600000	
75%	6.400000	3.200000	5.100000	1.800000	12.800000	
max	7.900000	4.100000	6.900000	2.500000	15.800000	

Outlier Detection and Removal

```
data.boxplot(figsize=(12,8))

# Here box and whisker plot is used to visualize any outliers present in the data.
```

<AxesSubplot:>




```
# applying log transformation
data['sepal diameter'] = np.log10(data['sepal diameter'])
data['sepal depth'] = np.log10(data['sepal depth'])
data['petal diameter'] = np.log10(data['petal diameter'])
data['petal depth'] = np.log10(data['petal depth'])
data.describe()
```

	sepal depth	sepal diameter	petal depth	petal diameter	sepal length	sep
count	150.000000	150.000000	150.000000	150.000000	150.000000	

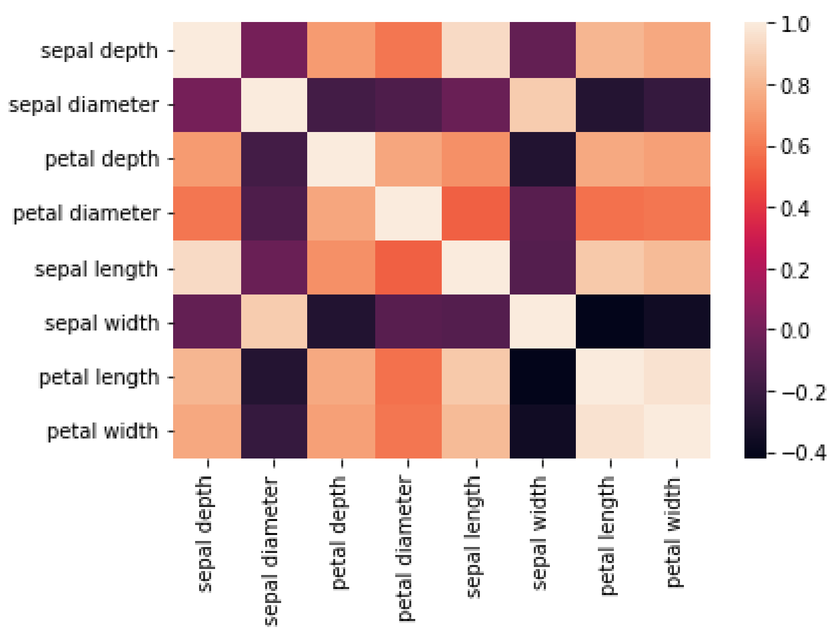
```
def remove_outlier(df_in, col_name):
    q1 = df_in[col_name].quantile(0.25)
    q3 = df_in[col_name].quantile(0.75)
    iqr = q3-q1 #Interquartile range
    fence_low  = q1-1.5*iqr
    fence_high = q3+1.5*iqr
    df_out = df_in.loc[(df_in[col_name] > fence_low) & (df_in[col_name] < fence_high)]
    return df_out

for i in (['petal depth','petal diameter','sepal depth','sepal diameter']):
    data=remove_outlier(data,i)
data.describe()
```

```
# After performing log transformation or outlier removal on this data, Log tranformati
# approach in this case as it does not reduce the size of data while minimizing the ou
```

Correlation and removing correlated columns

```
import seaborn as sns
sns.heatmap(data.corr())
```

```
<AxesSubplot:>
```




```
# Judging by this heatmap, we can see that most of the areas in heat map are towards c
# However the minimum correlation does not exceed a correlation of -0.5. So judging by
```

```
# correlation between the data. However we may still need further analysis to find out
# such as petal diameter and width may be highly related based on measuring criterias.
```