data-science-practicals-no-2

March 6, 2024

[119]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

[120]:
```
df=pd.read_csv("datasets/diabetes_modified.csv")
```

[121]:
```
df.head(2)
```

[121]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0          6.0    148.0           72.0           35.0      0.0  33.6
1          1.0     85.0           66.0           29.0      0.0  26.6

   DiabetesPedigreeFunction   Age       Outcome
0                     0.627  50.0      Diabetic
1                       NaN  31.0  Non-Diabetic
```

[122]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               762 non-null    float64
 1   Glucose                   751 non-null    float64
 2   BloodPressure             751 non-null    float64
 3   SkinThickness             753 non-null    float64
 4   Insulin                   765 non-null    float64
 5   BMI                       739 non-null    float64
 6   DiabetesPedigreeFunction  739 non-null    float64
 7   Age                       766 non-null    float64
 8   Outcome                   768 non-null    object
dtypes: float64(8), object(1)
memory usage: 54.1+ KB
```

[123]:
```
df.describe()
```

[123]:

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   762.000000  751.000000     751.000000     753.000000  765.000000
mean      3.824147  120.719041      69.102530      20.540505   79.905882
std       3.360596   31.958175      19.282846      15.912954  115.431340
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   29.000000
75%       6.000000  140.000000      80.000000      32.000000  128.000000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age
count  739.000000                739.000000  766.000000
mean    32.032882                  0.471766   33.227154
std      7.901092                  0.326533   11.755153
min      0.000000                  0.078000   21.000000
25%     27.350000                  0.245000   24.000000
50%     32.300000                  0.375000   29.000000
75%     36.600000                  0.621500   41.000000
max     67.100000                  2.420000   81.000000
```

[124]:

```
df.shape
```

[124]:

```
(768, 9)
```

[125]:

```
df.dtypes
```

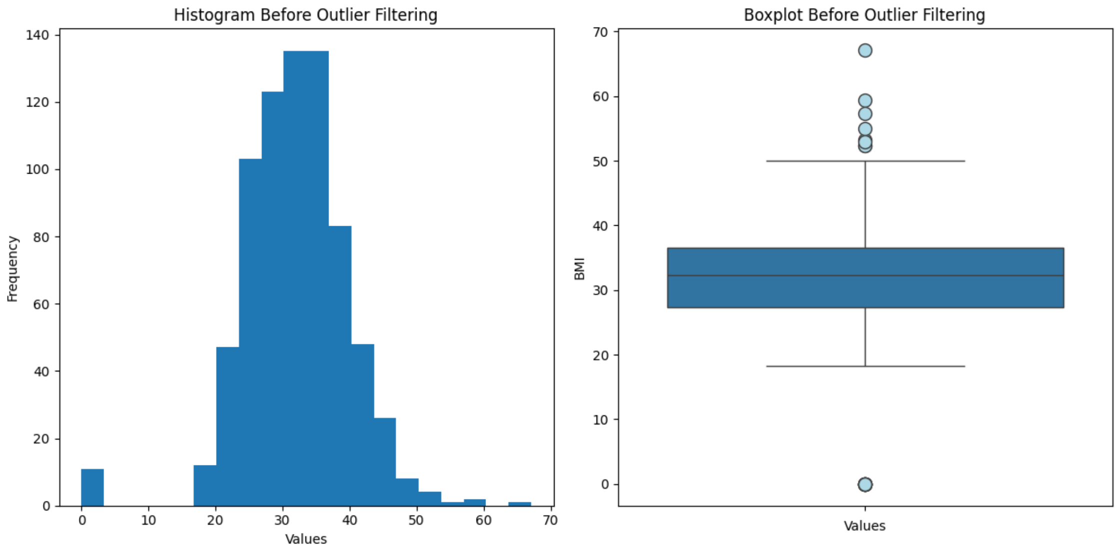
[125]:

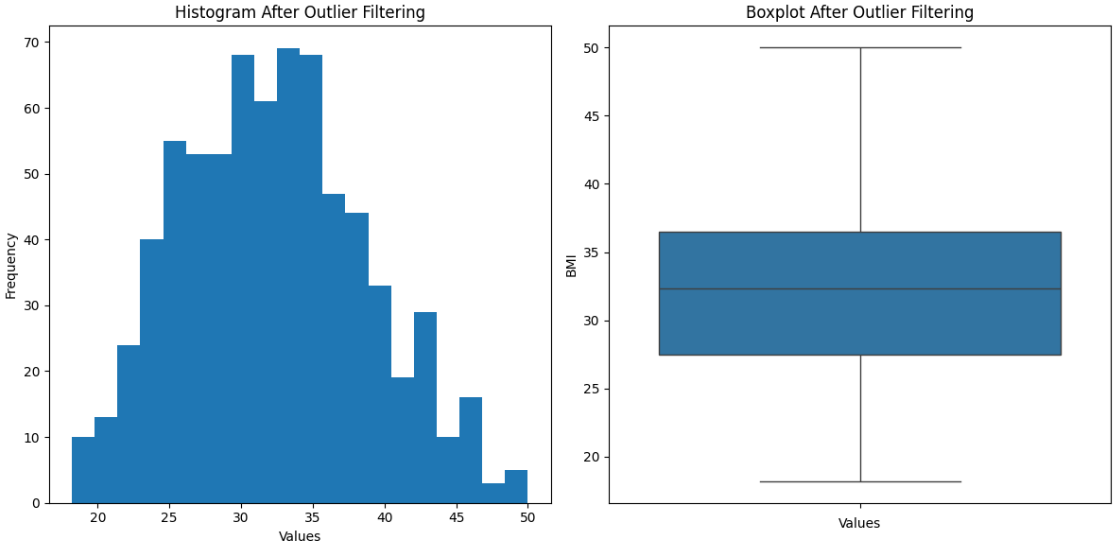
```
Pregnancies                 float64
Glucose                     float64
BloodPressure               float64
SkinThickness               float64
Insulin                     float64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                         float64
Outcome                      object
dtype: object
```

1 outlier

[126]:

```
def plot_histogram_and_boxplot(data, column_name, title_suffix):
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.hist(data[column_name], bins=20)
    plt.title(f'Histogram {title_suffix}')
    plt.xlabel('Values')
    plt.ylabel('Frequency')
    plt.subplot(1, 2, 2)
    sns.boxplot(y=data[column_name],
                flierprops=dict(marker='o', markerfacecolor='lightblue',␣
 ↪markersize=10))
    plt.title(f'Boxplot {title_suffix}')
    plt.xlabel('Values')

    plt.tight_layout()
    plt.show()

def remove_outliers_iqr(data, column_name):
    Q1 = data[column_name].quantile(0.25)
    Q3 = data[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data[column_name] >= lower_bound) & (data[column_name] <=␣
 ↪upper_bound)]


# Plots before outlier removal
plot_histogram_and_boxplot(df, 'BMI', 'Before Outlier Filtering')

# Remove outliers using IQR method
df_filtered = remove_outliers_iqr(df, 'BMI')

# Plots after outlier removal
plot_histogram_and_boxplot(df_filtered, 'BMI', 'After Outlier Filtering')
```



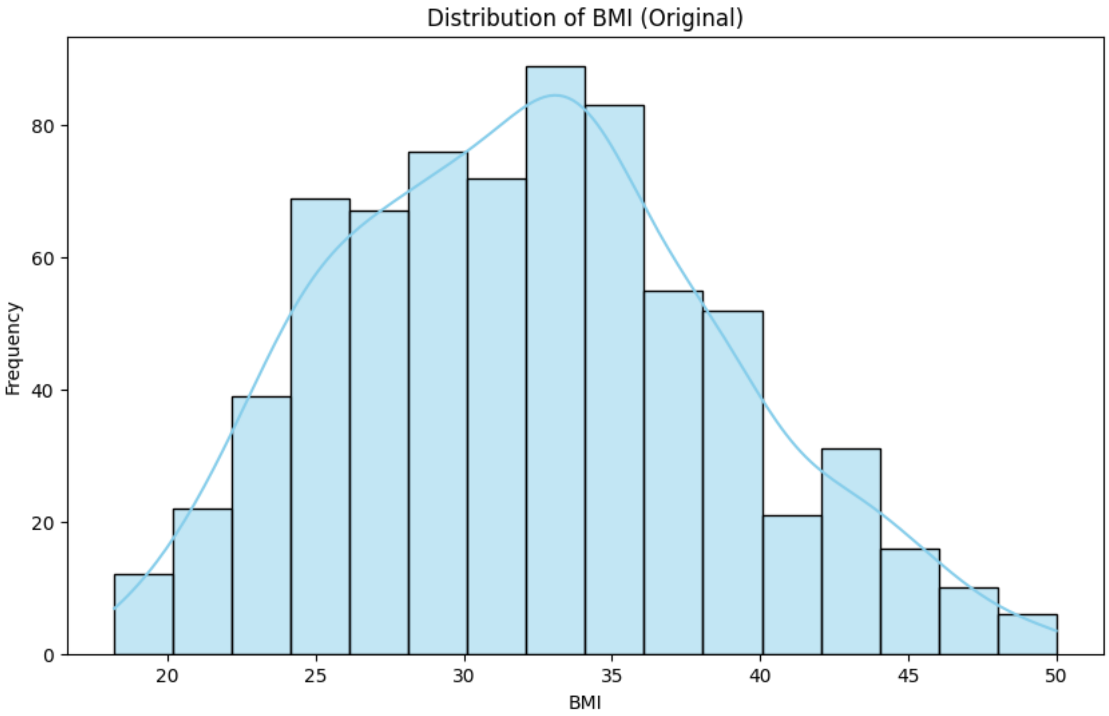


1.0.1 Transformation

[127]:
```
# skewness
original_skewness = df_filtered['BMI'].skew()
print("Original skewness of 'BMI' variable:", original_skewness)
```

```
Original skewness of 'BMI' variable: 0.2545074516989331
```

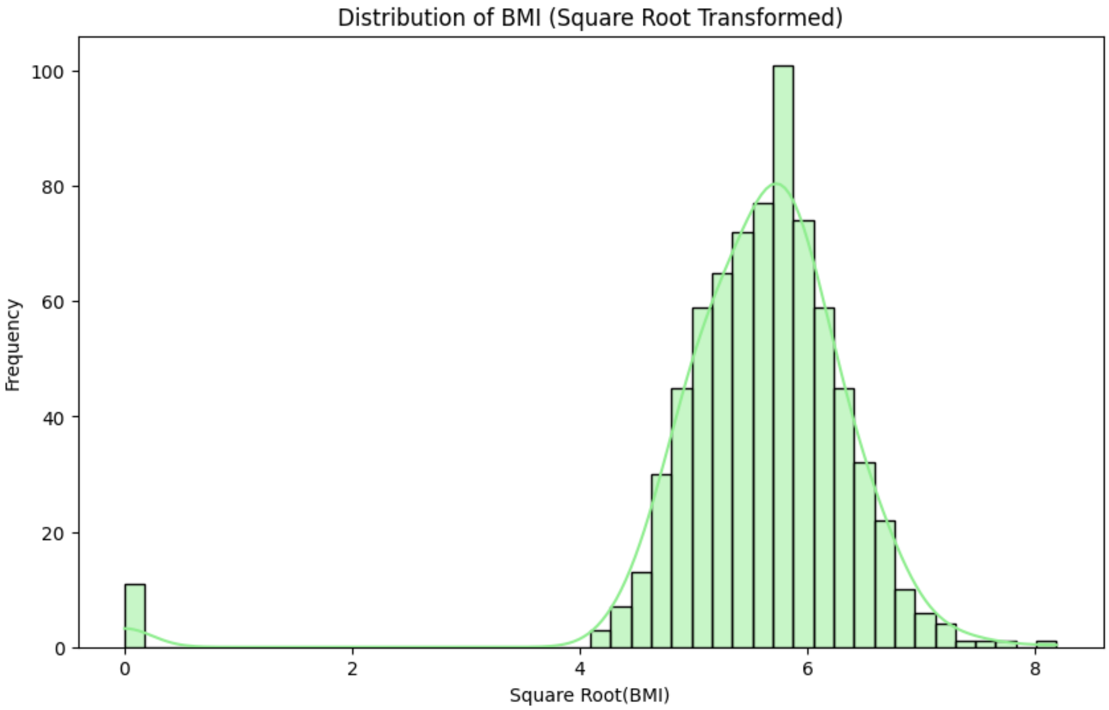
[128]:
```
# Plot
plt.figure(figsize=(10, 6))
sns.histplot(df_filtered['BMI'], kde=True, color='skyblue')
plt.title('Distribution of BMI (Original)')
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.show()
```



[129]:
```
# square root transformation
df['BMI_sqrt'] = np.sqrt(df['BMI'])

# skewness after transformation
transformed_skewness = df['BMI_sqrt'].skew()
print("Skewness of 'BMI' variable after square root transformation:",␣
 ↪transformed_skewness)
```

```
Skewness of 'BMI' variable after square root transformation: -3.283161243908821
```

[130]:
```
# Plot
plt.figure(figsize=(10, 6))
sns.histplot(df['BMI_sqrt'], kde=True, color='lightgreen')
plt.title('Distribution of BMI (Square Root Transformed)')
plt.xlabel('Square Root(BMI)')
plt.ylabel('Frequency')
plt.show()
```


Distribution of BMI (Square Root Transformed)
Frequency
0
20
40
60
80
100
0
2
4
6
8
Square Root(BMI)