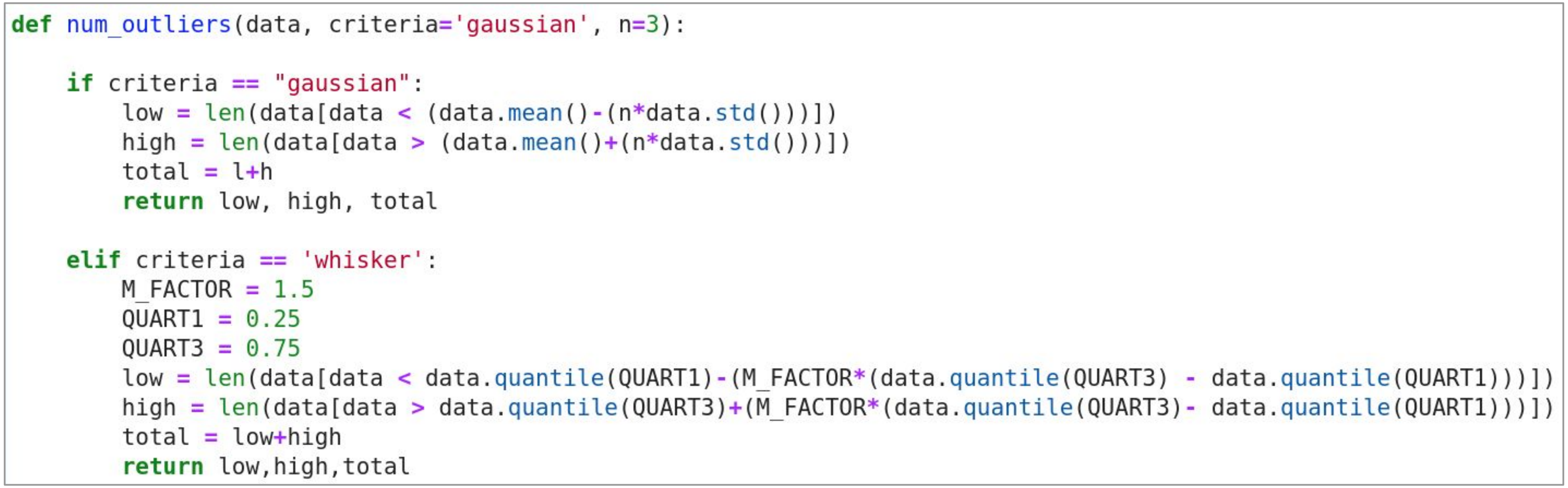
Effective Code Commenting

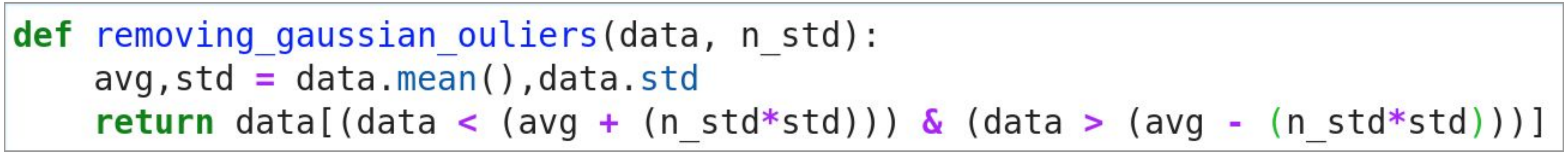
Analytics Vidhya

Code Commenting



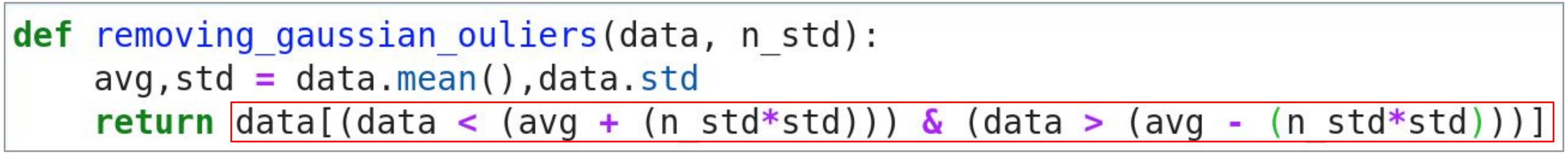
```python
def num_outliers(data, criteria='gaussian', n=3):

    if criteria == "gaussian":
        low = len(data[data < (data.mean()-(n*data.std()))])
        high = len(data[data > (data.mean()+(n*data.std()))])
        total = l+h
        return low, high, total

    elif criteria == 'whisker':
        M_FACTOR = 1.5
        QUART1 = 0.25
        QUART3 = 0.75
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        total = low+high
        return low,high,total
```

Code Commenting



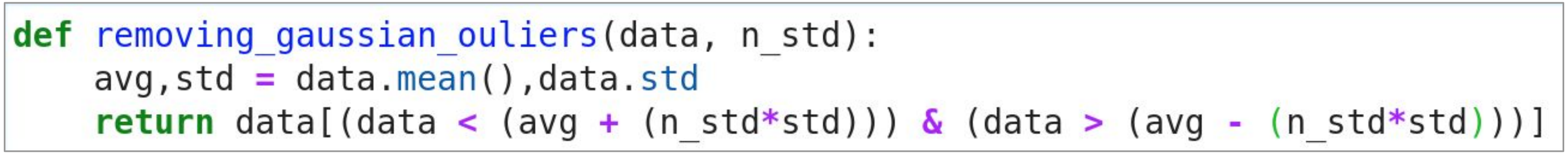
```python
def removing_gaussian_ouliers(data, n_std):
    avg,std = data.mean(),data.std
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Code Commenting

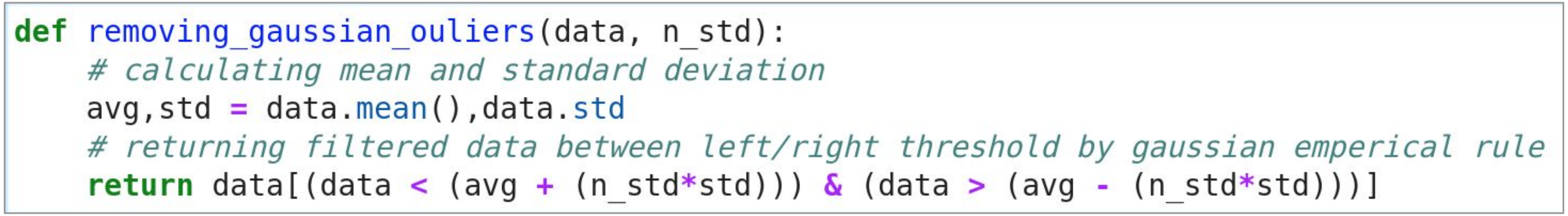


```python
def removing_gaussian_ouliers(data, n_std):
    avg,std = data.mean(),data.std
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Code Commenting

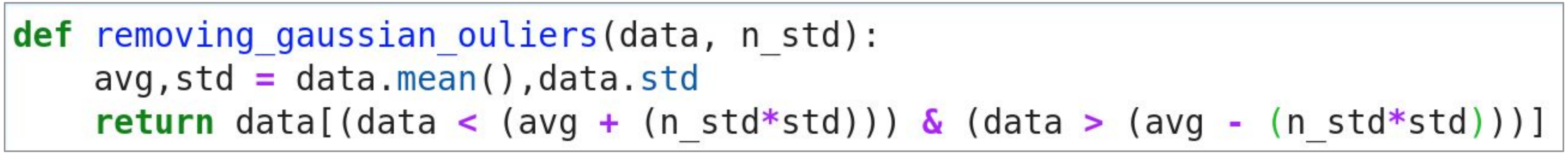


```python
def removing_gaussian_ouliers(data, n_std):
    avg,std = data.mean(),data.std
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

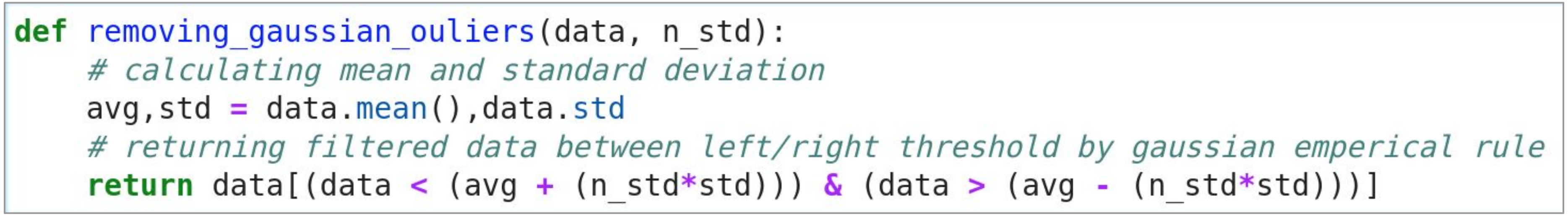


```python
def removing_gaussian_ouliers(data, n_std):
    # calculating mean and standard deviation
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Code Commenting



```python
def removing_gaussian_ouliers(data, n_std):
    avg,std = data.mean(),data.std
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```



```python
def removing_gaussian_ouliers(data, n_std):
    # calculating mean and standard deviation
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Redundant and time taking?

Code Commenting

```python
def removing_gaussian_ouliers(data, n_std):
    avg,std = data.mean(),data.std
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

```python
def removing_gaussian_ouliers(data, n_std):
    # calculating mean and standard deviation
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Redundant and time taking?

Yes and No

Code Commenting

When Writing Code:

- Code should be self explanatory and readable

Code Commenting

When Writing Code:

- Code should be self explanatory and readable
- If too complex, add comment

Code Commenting

When Writing Code:

- Code should be self explanatory and readable
- If too complex, add comment
- Comments increase length

Code Commenting

When Writing Code:

- Code should be self explanatory and readable
- If too complex, add comment
- Comments increase length
- Time taking

Code Commenting

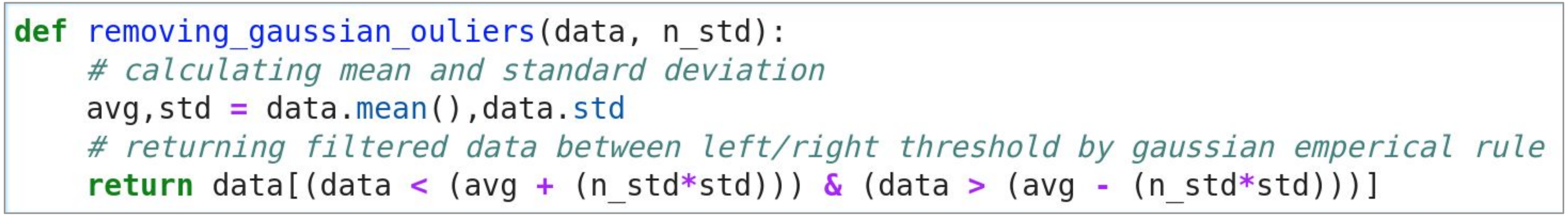
When Writing Code:

- Code should be self explanatory and readable
- If too complex, add comment
- Comments increase length
- Time taking

Comments should be a last resort, but necessary if code is complex

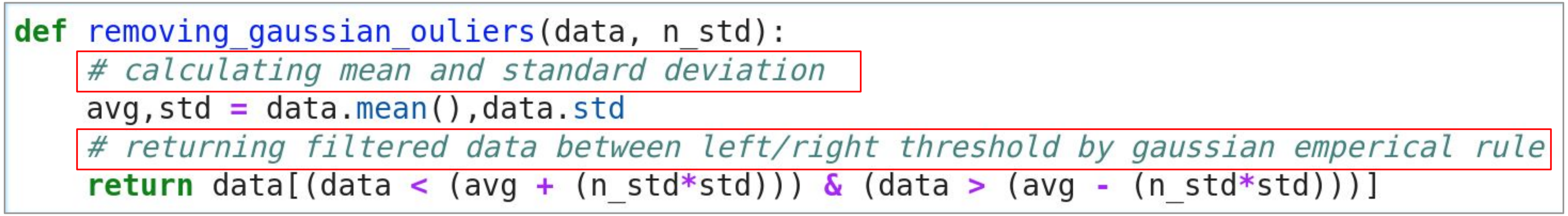
Analytics Vidhya

Code Commenting



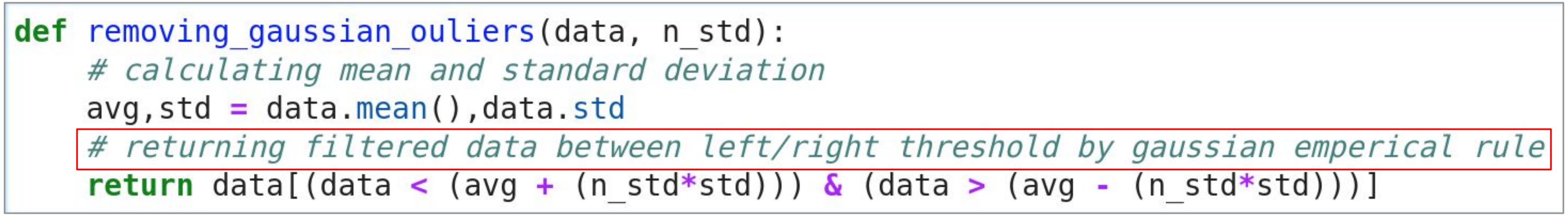
```python
def removing_gaussian_ouliers(data, n_std):
    # calculating mean and standard deviation
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Code Commenting



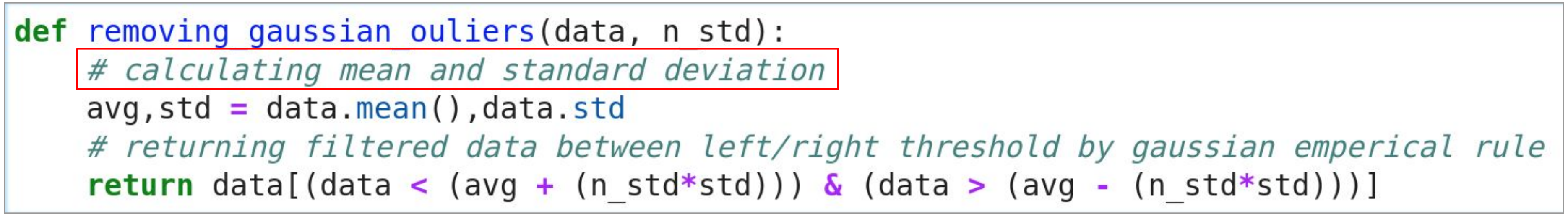
```python
def removing_gaussian_ouliers(data, n_std):
    # calculating mean and standard deviation
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Code Commenting



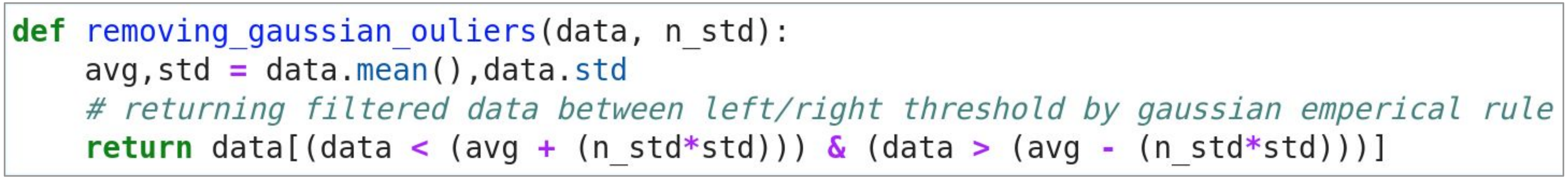
```
def removing_gaussian_ouliers(data, n_std):
    # calculating mean and standard deviation
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

Code Commenting



```python
def removing_gaussian_ouliers(data, n_std):
    # calculating mean and standard deviation
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

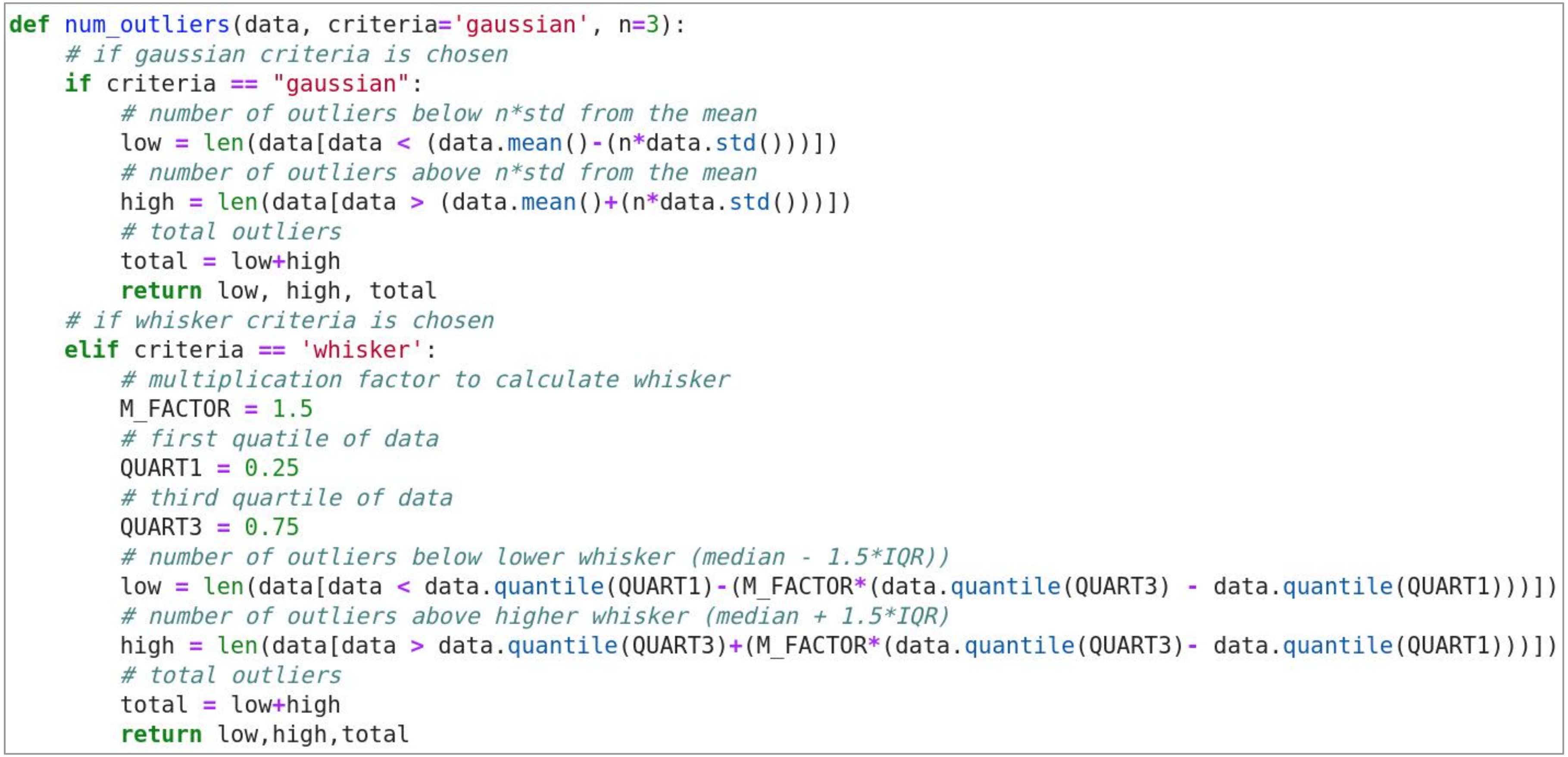
Code Commenting



```
def removing_gaussian_ouliers(data, n_std):
    avg,std = data.mean(),data.std
    # returning filtered data between left/right threshold by gaussian emperical rule
    return data[(data < (avg + (n_std*std))) & (data > (avg - (n_std*std)))]
```

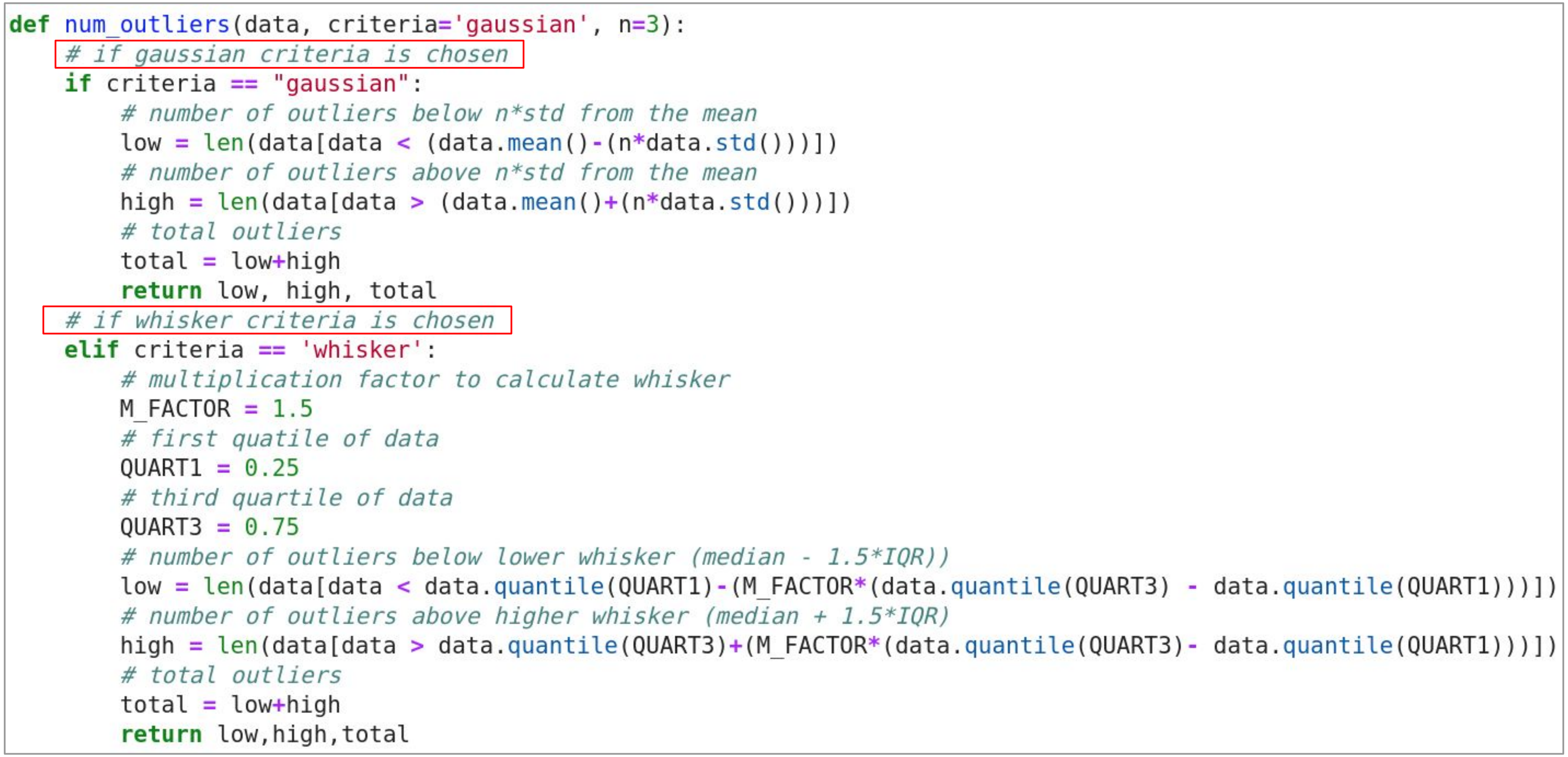
Redundant Commenting

Redundant Commenting



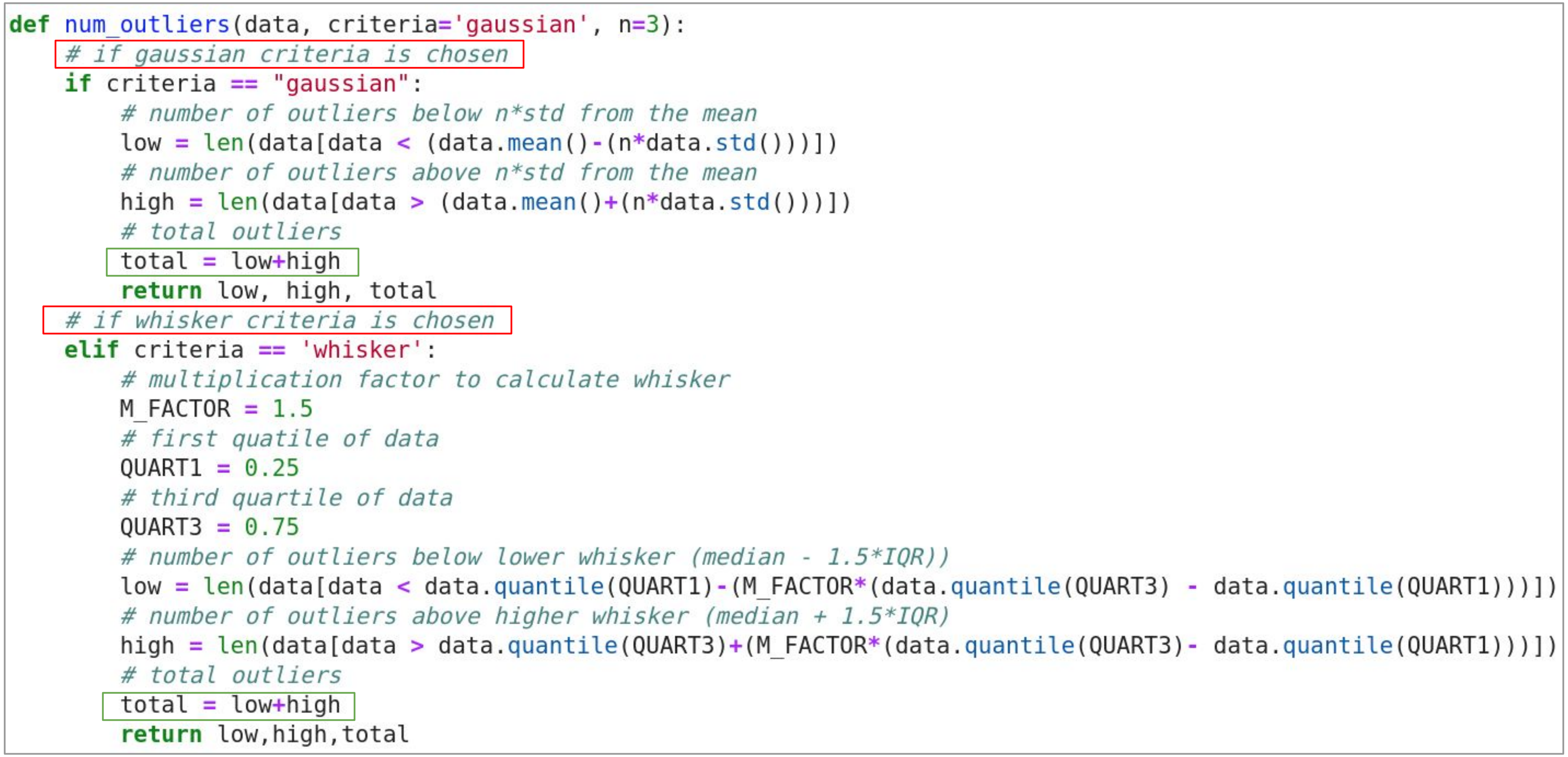
```python
def num_outliers(data, criteria='gaussian', n=3):
    # if gaussian criteria is chosen
    if criteria == "gaussian":
        # number of outliers below n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        # number of outliers above n*std from the mean
        high = len(data[data > (data.mean()+(n*data.std()))])
        # total outliers
        total = low+high
        return low, high, total
    # if whisker criteria is chosen
    elif criteria == 'whisker':
        # multiplication factor to calculate whisker
        M_FACTOR = 1.5
        # first quatile of data
        QUART1 = 0.25
        # third quartile of data
        QUART3 = 0.75
        # number of outliers below lower whisker (median - 1.5*IQR))
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        # number of outliers above higher whisker (median + 1.5*IQR)
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        # total outliers
        total = low+high
        return low,high,total
```

Redundant Commenting



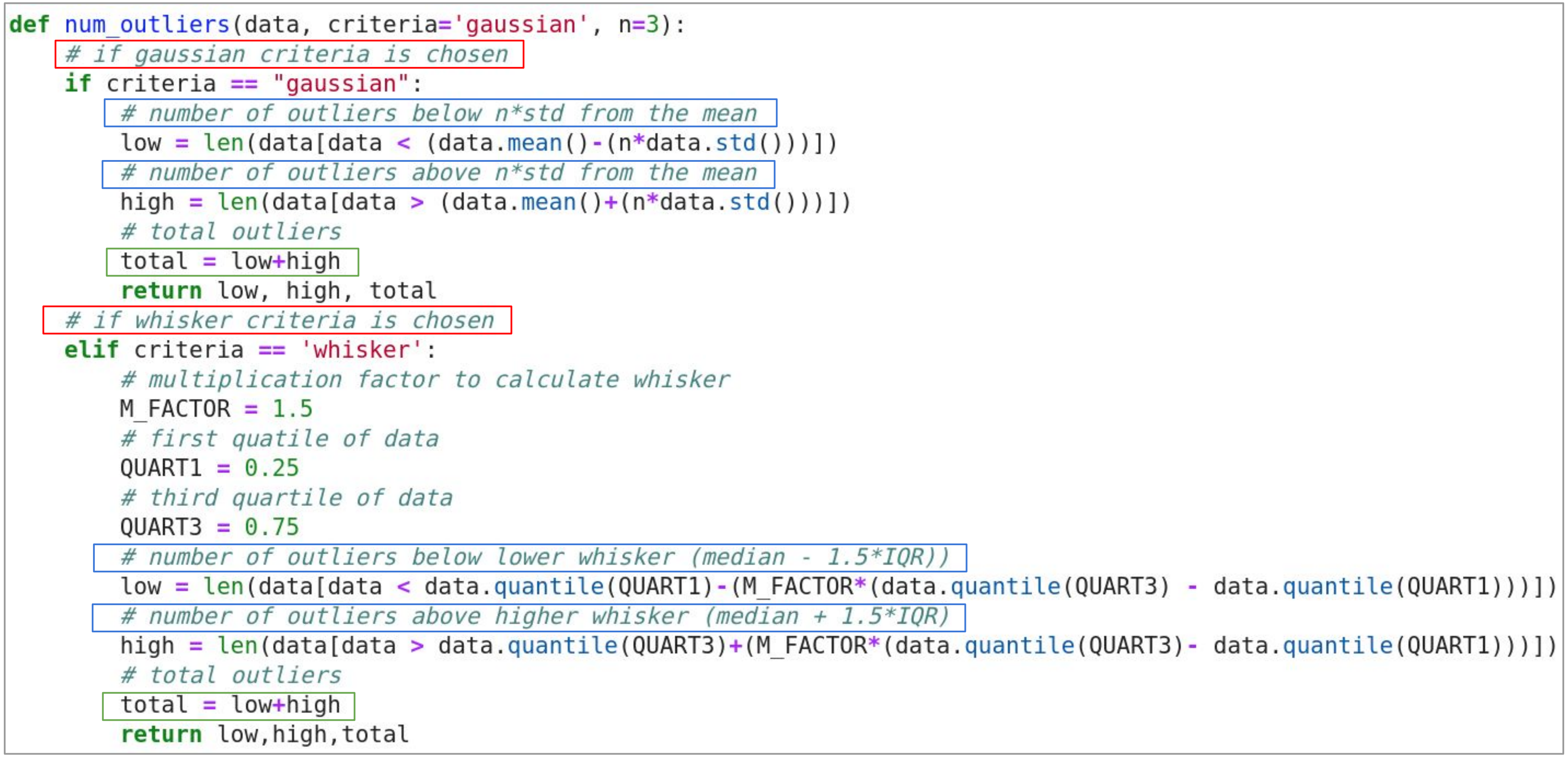
```python
def num_outliers(data, criteria='gaussian', n=3):
    # if gaussian criteria is chosen
    if criteria == "gaussian":
        # number of outliers below n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        # number of outliers above n*std from the mean
        high = len(data[data > (data.mean()+(n*data.std()))])
        # total outliers
        total = low+high
        return low, high, total
    # if whisker criteria is chosen
    elif criteria == 'whisker':
        # multiplication factor to calculate whisker
        M_FACTOR = 1.5
        # first quatile of data
        QUART1 = 0.25
        # third quartile of data
        QUART3 = 0.75
        # number of outliers below lower whisker (median - 1.5*IQR))
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        # number of outliers above higher whisker (median + 1.5*IQR)
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        # total outliers
        total = low+high
        return low,high,total
```

Redundant Commenting



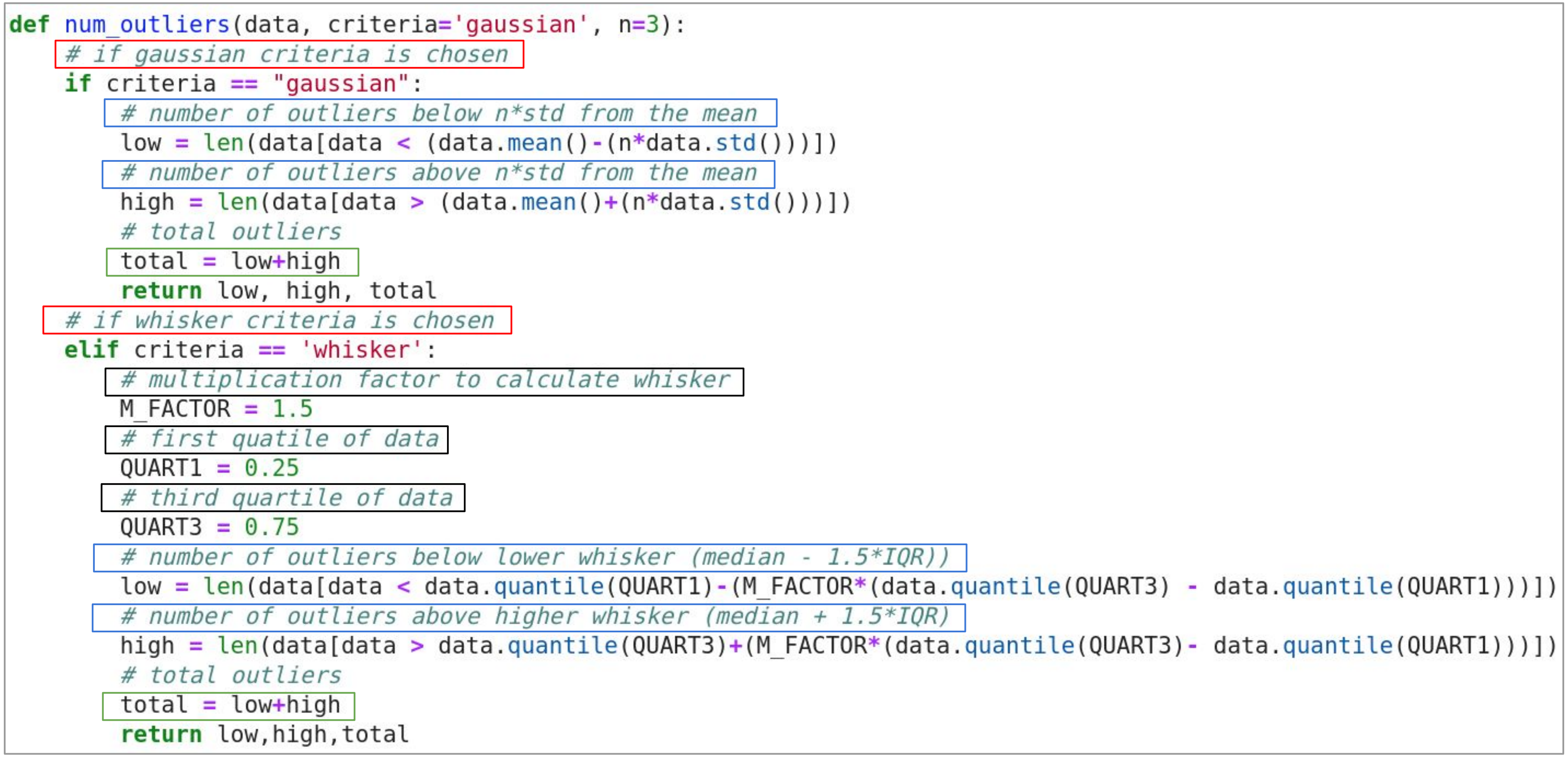
```
def num_outliers(data, criteria='gaussian', n=3):
    # if gaussian criteria is chosen
    if criteria == "gaussian":
        # number of outliers below n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        # number of outliers above n*std from the mean
        high = len(data[data > (data.mean()+(n*data.std()))])
        # total outliers
        total = low+high
        return low, high, total
    # if whisker criteria is chosen
    elif criteria == 'whisker':
        # multiplication factor to calculate whisker
        M_FACTOR = 1.5
        # first quatile of data
        QUART1 = 0.25
        # third quartile of data
        QUART3 = 0.75
        # number of outliers below lower whisker (median - 1.5*IQR))
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1))))])
        # number of outliers above higher whisker (median + 1.5*IQR)
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1))))])
        # total outliers
        total = low+high
        return low,high,total
```

Redundant Commenting



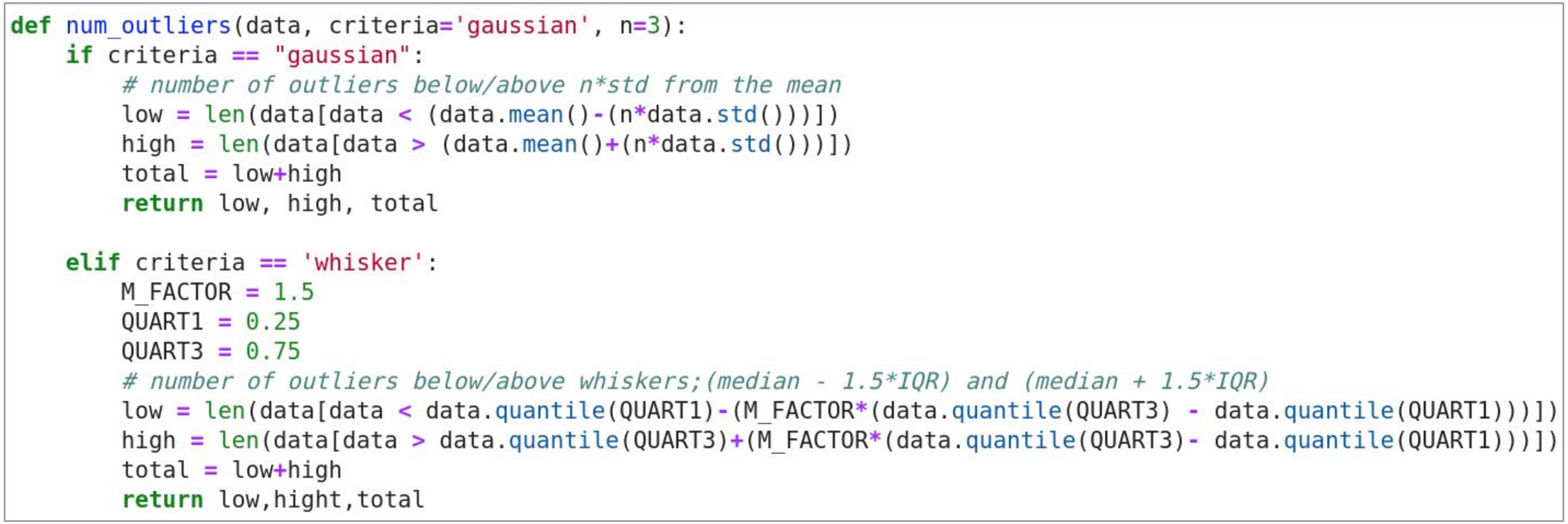
```
def num_outliers(data, criteria='gaussian', n=3):
    # if gaussian criteria is chosen
    if criteria == "gaussian":
        # number of outliers below n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        # number of outliers above n*std from the mean
        high = len(data[data > (data.mean()+(n*data.std()))])
        # total outliers
        total = low+high
        return low, high, total
    # if whisker criteria is chosen
    elif criteria == 'whisker':
        # multiplication factor to calculate whisker
        M_FACTOR = 1.5
        # first quatile of data
        QUART1 = 0.25
        # third quartile of data
        QUART3 = 0.75
        # number of outliers below lower whisker (median - 1.5*IQR))
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        # number of outliers above higher whisker (median + 1.5*IQR)
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        # total outliers
        total = low+high
        return low,high,total
```

Redundant Commenting



```python
def num_outliers(data, criteria='gaussian', n=3):
    # if gaussian criteria is chosen
    if criteria == "gaussian":
        # number of outliers below n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        # number of outliers above n*std from the mean
        high = len(data[data > (data.mean()+(n*data.std()))])
        # total outliers
        total = low+high
        return low, high, total
    # if whisker criteria is chosen
    elif criteria == 'whisker':
        # multiplication factor to calculate whisker
        M_FACTOR = 1.5
        # first quatile of data
        QUART1 = 0.25
        # third quartile of data
        QUART3 = 0.75
        # number of outliers below lower whisker (median - 1.5*IQR))
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        # number of outliers above higher whisker (median + 1.5*IQR)
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        # total outliers
        total = low+high
        return low,high,total
```

Effective Commenting



```python
def num_outliers(data, criteria='gaussian', n=3):
    if criteria == "gaussian":
        # number of outliers below/above n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        high = len(data[data > (data.mean()+(n*data.std()))])
        total = low+high
        return low, high, total

    elif criteria == 'whisker':
        M_FACTOR = 1.5
        QUART1 = 0.25
        QUART3 = 0.75
        # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        total = low+high
        return low,hight,total
```

Analytics Vidhya

Thank You

Analytics Vidhya