Find and Remove Outliers in Credit Card Fraud Detection Dataset

Please download the data from https://www.kaggle.com/dalpozz/creditcardfraud/data (https://www.kaggle.com/dalpozz/creditcardfraud/data)

Info about data: it is a CSV file, contains 31 features, the last feature is used to classify the transaction whether it is a fraud or not

Information about data set

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. **Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.**

Given the class imbalance ratio, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be (http://mlg.ulb.ac.be)) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.neighbors import LocalOutlierFactor
```

In [2]:
```
data = pd.read_csv("creditcard.csv")
```

In [3]:
```
data.shape
```

Out[3]:
```
(284807, 31)
```

In [4]:
```
data.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141

5 rows × 31 columns

In [5]:
```
# sampling random 50000 points
data_50000 = data.sample(n = 50000)
```

In [6]:
```
data_50000.head()
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V2
74072	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	1.446847	...	0.018695	0.357714	-0.24163
37724	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	-1.197673	...	0.513156	1.328134	-0.46505
207494	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	0.405572	...	0.253490	0.800281	-0.09720
180930	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	0.525018	...	-0.145317	0.581410	-0.47992
271725	164713.0	-1.424896	1.073129	0.105897	-0.941510	0.437835	0.584503	0.103029	1.050873	-0.247711	...	0.094488	0.238311	-0.06067

5 rows × 31 columns

In [7]:
```
data_50000.to_csv("NewCreditCard.csv")
```

Detecting outliers for 'k' value 2

We have assumed that 50% of total points in our data set are outliers.

In [8]:
```
newData = pd.read_csv("NewCreditCard.csv")
```

In [9]:
```
newData.head()
```

Out[9]:

	Unnamed: 0	Time	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22	V23	
0	74072	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	...	0.018695	0.357714	-0.241636	-0
1	37724	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	...	0.513156	1.328134	-0.465055	-0
2	207494	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	...	0.253490	0.800281	-0.097207	0
3	180930	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	...	-0.145317	0.581410	-0.479921	-0
4	271725	164713.0	-1.424896	1.073129	0.105897	-0.941510	0.437835	0.584503	0.103029	1.050873	...	0.094488	0.238311	-0.060676	-0

5 rows × 32 columns

In [10]:
```
FinalData = newData.drop("Unnamed: 0", axis = 1)
```

In [11]:
```
FinalData.head()
```

Out[11]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	1.446847	...	0.018695	0.357714	-0.241636	-0.
1	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	-1.197673	...	0.513156	1.328134	-0.465055	-0.
2	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	0.405572	...	0.253490	0.800281	-0.097207	0.
3	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	0.525018	...	-0.145317	0.581410	-0.479921	-0.
4	164713.0	-1.424896	1.073129	0.105897	-0.941510	0.437835	0.584503	0.103029	1.050873	-0.247711	...	0.094488	0.238311	-0.060676	-0.

5 rows × 31 columns

In [12]:
```
FinalData.shape
```

Out[12]:
```
(50000, 31)
```

In [26]:
```
lof = LocalOutlierFactor(n_neighbors=2, algorithm='auto', metric='minkowski', p=2, metric_params=None, contamination=0.5,

outlierArray = lof.fit_predict(FinalData)

outlierArray
```

Out[26]:
```
array([-1, -1,  1, ...,  1,  1,  1])
```

Here, we got an array, where row corresponding to array element 1 in our dataset is an inlier and row corresponding to array element to -1 in our dataset is an oulier

In [27]:
```
len(outlierArray)
```

Out[27]:
```
50000
```

Calculating total number of outlier and inliers

In [28]:
```
countOutliers = 0
countInliers = 0
for i in range(50000):
    if outlierArray[i] == -1:
        countOutliers += 1
    else:
        countInliers += 1
print("Total number of outliers = "+str(countOutliers))
print("Total number of inliers = "+str(countInliers))
```

```
Total number of outliers = 25000
Total number of inliers = 25000
```

In [29]:
```
FinalData2 = FinalData.copy()
```

In [30]:
```
FinalData2.shape
```

Out[30]:
```
(50000, 31)
```

Removing Outliers

In [31]:
```
for i in range(50000):
    if outlierArray[i] == -1:
        FinalData.drop(i, inplace = True)
FinalData.head()
```

Out[31]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
2	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	0.405572	...	0.253490	0.800281	-0.097207	0.
3	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	0.525018	...	-0.145317	0.581410	-0.479921	-0.
6	154725.0	1.982318	-0.535051	-0.214848	0.521755	-0.983215	-0.640933	-0.727199	0.003512	1.439345	...	0.149974	0.598656	0.178110	0.
7	32724.0	-0.455484	0.670802	1.465590	-1.228577	-0.004430	-0.935596	0.801324	-0.214961	0.259080	...	-0.066009	-0.013992	-0.025550	0.
8	25097.0	0.328613	1.784994	-2.151250	1.984450	0.430838	-1.748658	0.782288	0.059741	0.801911	...	-0.209499	-0.406531	0.360340	-0.

5 rows × 31 columns

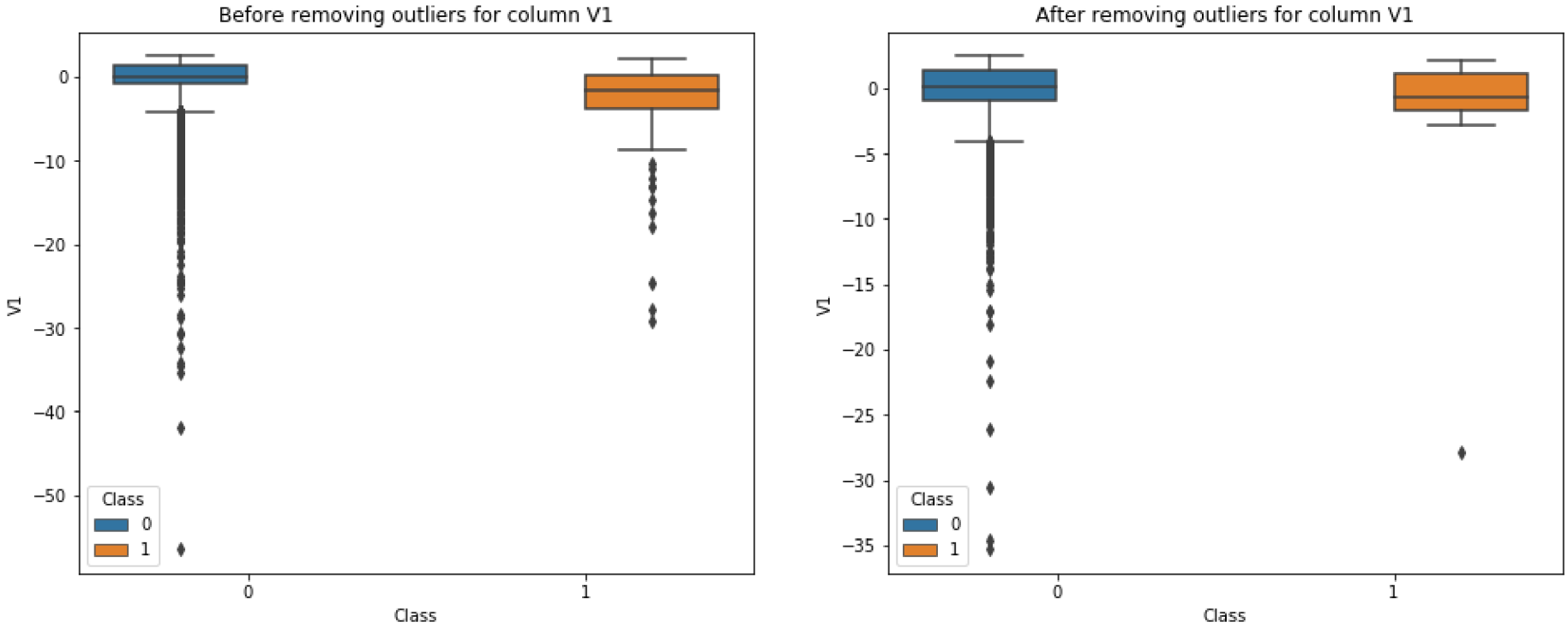
In [32]:
```
FinalData2.shape
```

Out[32]: (50000, 31)

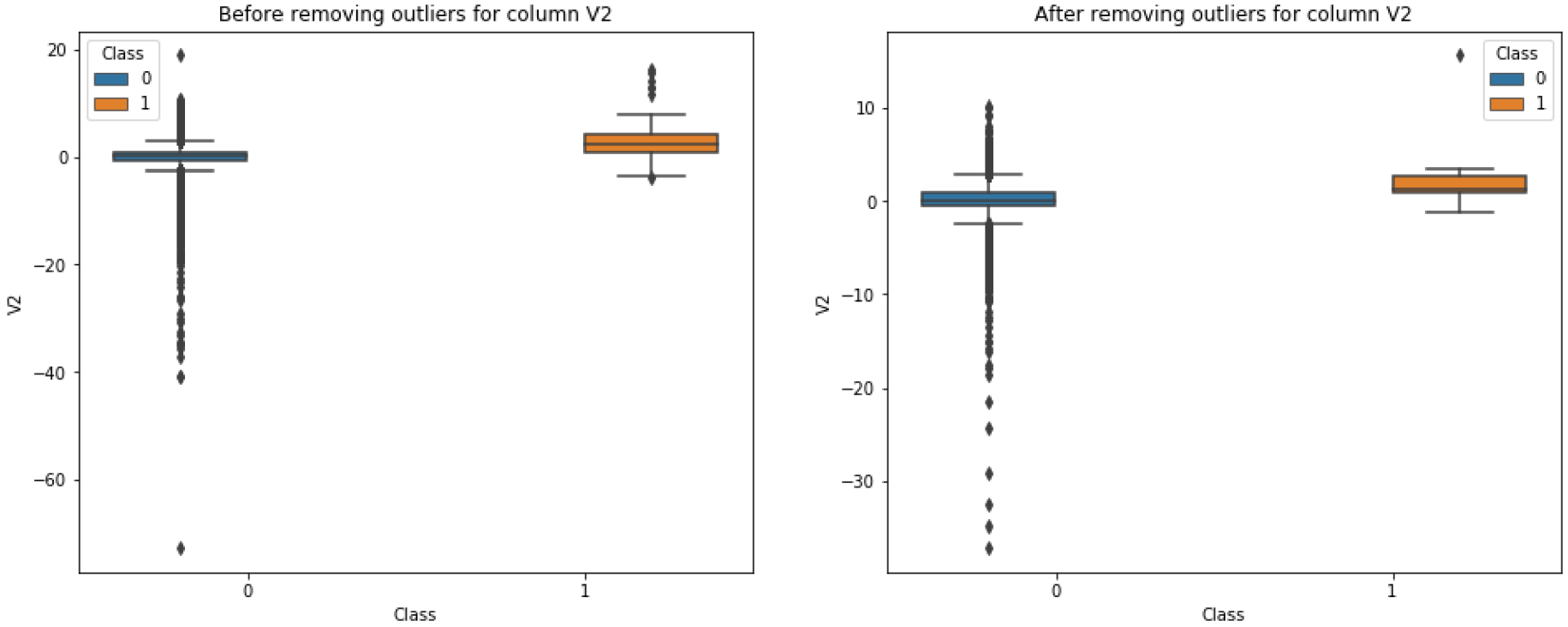
In [33]:
```
FinalData.shape
```

Out[33]: (25000, 31)

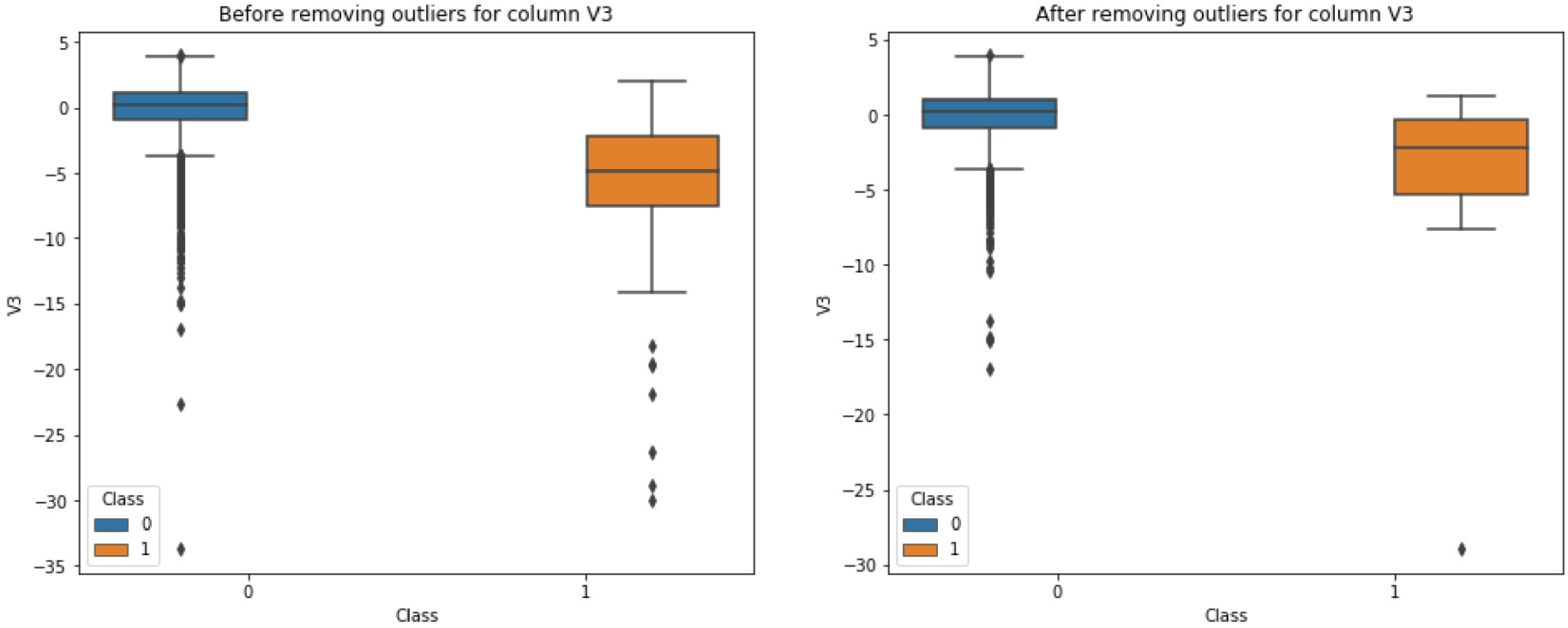
In [34]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V1")
ax = sns.boxplot(x="Class", y = "V1", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V1")
ax = sns.boxplot(x="Class", y = "V1", data= FinalData, hue = "Class")
```



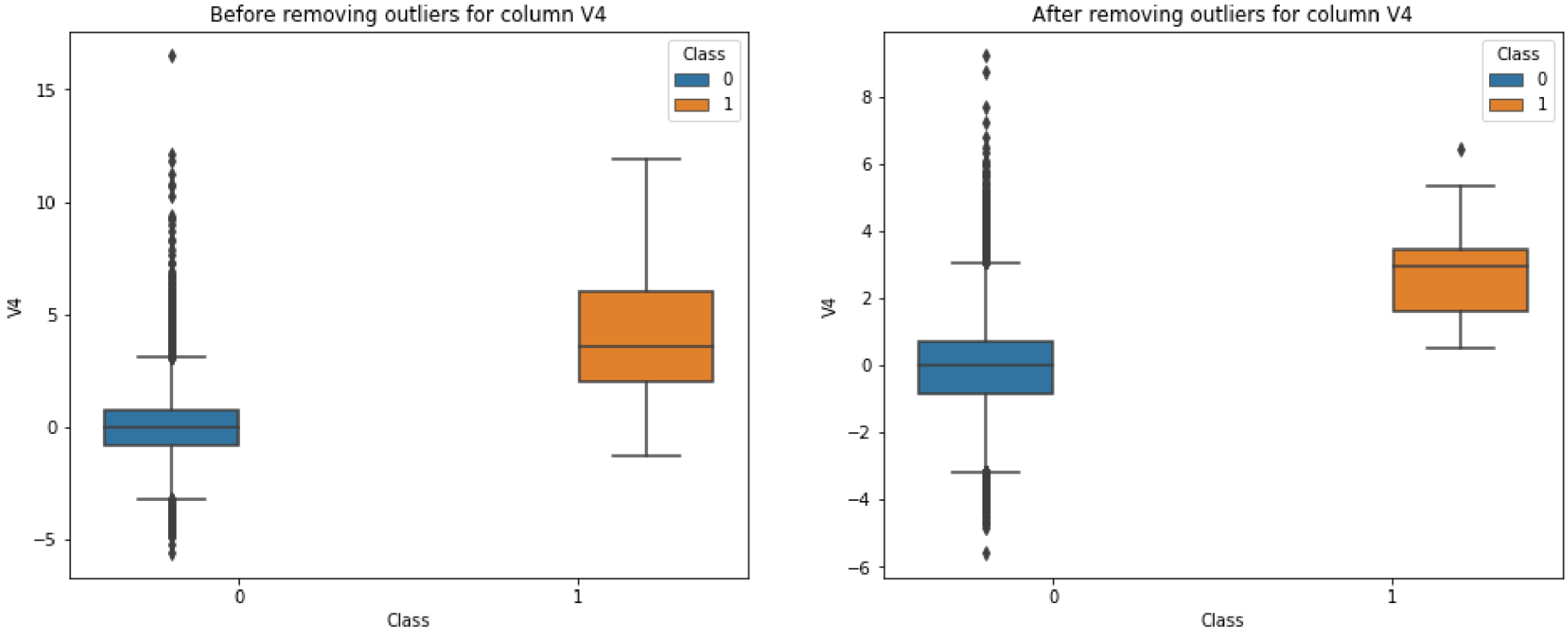
In [35]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V2")
ax = sns.boxplot(x="Class", y = "V2", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V2")
ax = sns.boxplot(x="Class", y = "V2", data= FinalData, hue = "Class")
```



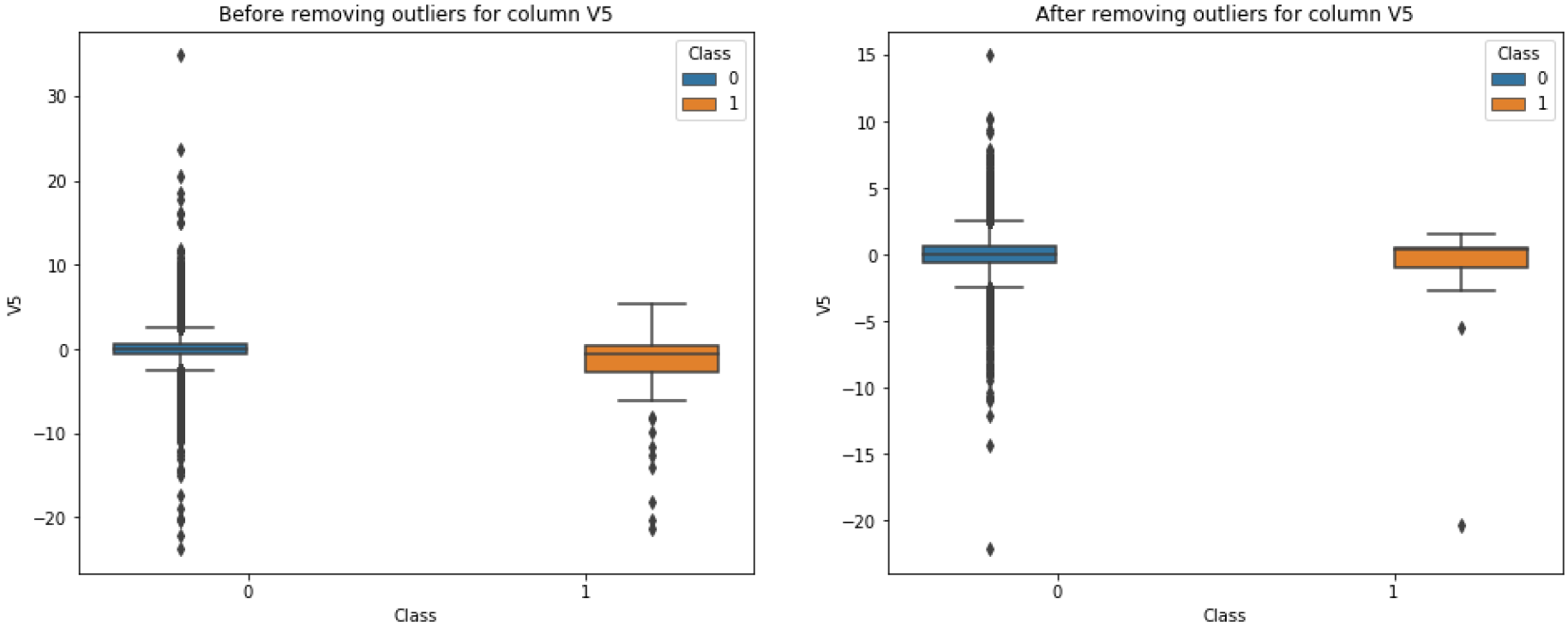
In [36]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V3")
ax = sns.boxplot(x="Class", y = "V3", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V3")
ax = sns.boxplot(x="Class", y = "V3", data= FinalData, hue = "Class")
```



In [37]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V4")
ax = sns.boxplot(x="Class", y = "V4", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V4")
ax = sns.boxplot(x="Class", y = "V4", data= FinalData, hue = "Class")
```



In [38]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V5")
ax = sns.boxplot(x="Class", y = "V5", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V5")
ax = sns.boxplot(x="Class", y = "V5", data= FinalData, hue = "Class")
```



It can easily be observed in all of the above boxplots corresponding to columns V1, V2, V3, V4, V5, that most of the outliers for points belongs to class 1 has been removed. Furthermore, many of the outliers for points belong to class 0 has also been removed. For example for class 0 in plot 1 for column V1 all of the outliers less than -35 have been removed. Similarly, for class 0 in plot 3 for column V3 all the outliers less than -18 have been removed. Similarly, for class 0 in plot 5 for column V5 all of the outliers greater than 15 have been removed.

Detecting outliers for 'k' value 3

We have assumed that 50% of total points in our data set are outliers.

In [39]:
```
newData = pd.read_csv("NewCreditCard.csv")

newData.head()
```

Out[39]:

	Unnamed: 0	Time	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22	V23	
0	74072	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	...	0.018695	0.357714	-0.241636	-0
1	37724	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	...	0.513156	1.328134	-0.465055	-0
2	207494	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	...	0.253490	0.800281	-0.097207	0
3	180930	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	...	-0.145317	0.581410	-0.479921	-0
4	271725	164713.0	-1.424896	1.073129	0.105897	-0.941510	0.437835	0.584503	0.103029	1.050873	...	0.094488	0.238311	-0.060676	-0

5 rows × 32 columns

In [40]:
```
FinalData = newData.drop("Unnamed: 0", axis = 1)

FinalData.head()
```

Out[40]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	1.446847	...	0.018695	0.357714	-0.241636	-0.
1	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	-1.197673	...	0.513156	1.328134	-0.465055	-0.
2	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	0.405572	...	0.253490	0.800281	-0.097207	0.
3	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	0.525018	...	-0.145317	0.581410	-0.479921	-0.
4	164713.0	-1.424896	1.073129	0.105897	-0.941510	0.437835	0.584503	0.103029	1.050873	-0.247711	...	0.094488	0.238311	-0.060676	-0.

5 rows × 31 columns

In [41]:
```
FinalData.shape
```

Out[41]:
```
(50000, 31)
```

In [42]:
```
lof = LocalOutlierFactor(n_neighbors=3, algorithm='auto', metric='minkowski', p=2, metric_params=None, contamination=0.5,

outlierArray = lof.fit_predict(FinalData)

outlierArray
```

Out[42]:
```
array([ 1,  1,  1, ..., -1, -1, -1])
```

Here, we got an array, where row corresponding to array element 1 in our dataset is an inlier and row corresponding to array element to -1 in our dataset is an oulier

In [43]:
```
len(outlierArray)
```

Out[43]:
```
50000
```

Calculating total number of outlier and inliers

In [44]:
```
countOutliers = 0
countInliers = 0
for i in range(50000):
    if outlierArray[i] == -1:
        countOutliers += 1
    else:
        countInliers += 1
print("Total number of outliers = "+str(countOutliers))
print("Total number of inliers = "+str(countInliers))
```

```
Total number of outliers = 25000
Total number of inliers = 25000
```

In [45]:
```
FinalData2 = FinalData.copy()
```

In [46]:
```
FinalData2.shape
```

Out[46]:
```
(50000, 31)
```

Removing Outliers

In [47]:
```
for i in range(50000):
    if outlierArray[i] == -1:
        FinalData.drop(i, inplace = True)
FinalData.head()
```

Out[47]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	1.446847	...	0.018695	0.357714	-0.241636	-0
1	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	-1.197673	...	0.513156	1.328134	-0.465055	-0
2	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	0.405572	...	0.253490	0.800281	-0.097207	0
6	154725.0	1.982318	-0.535051	-0.214848	0.521755	-0.983215	-0.640933	-0.727199	0.003512	1.439345	...	0.149974	0.598656	0.178110	0
7	32724.0	-0.455484	0.670802	1.465590	-1.228577	-0.004430	-0.935596	0.801324	-0.214961	0.259080	...	-0.066009	-0.013992	-0.025550	0

5 rows × 31 columns

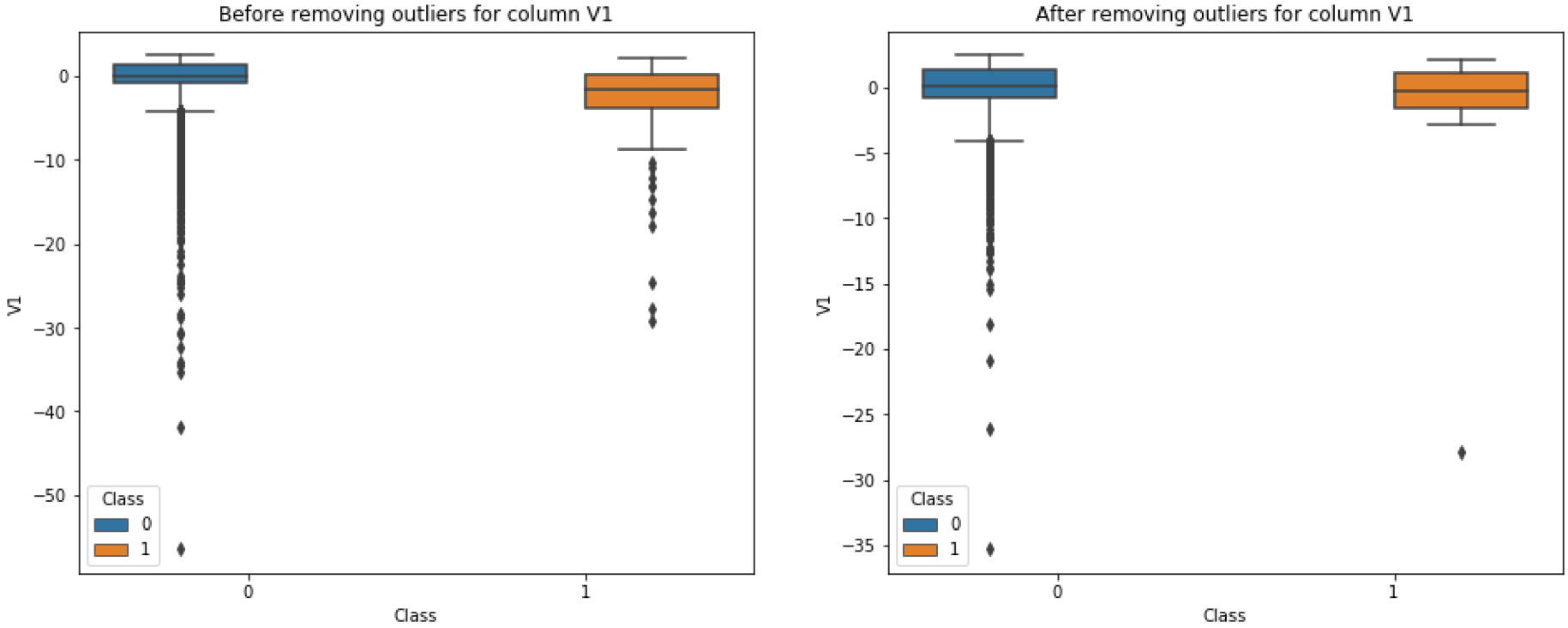
In [48]:
```
FinalData2.shape
```

Out[48]:
```
(50000, 31)
```

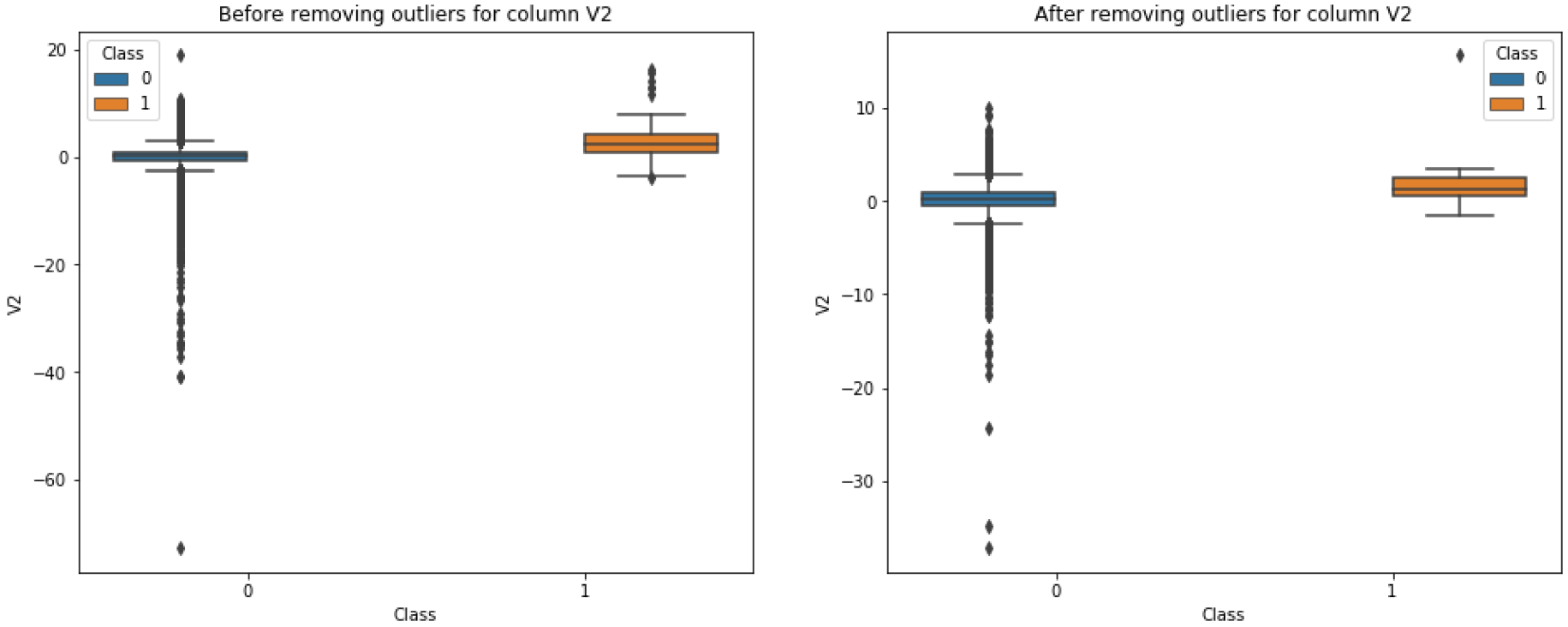
In [49]:
```
FinalData.shape
```

Out[49]:
```
(25000, 31)
```

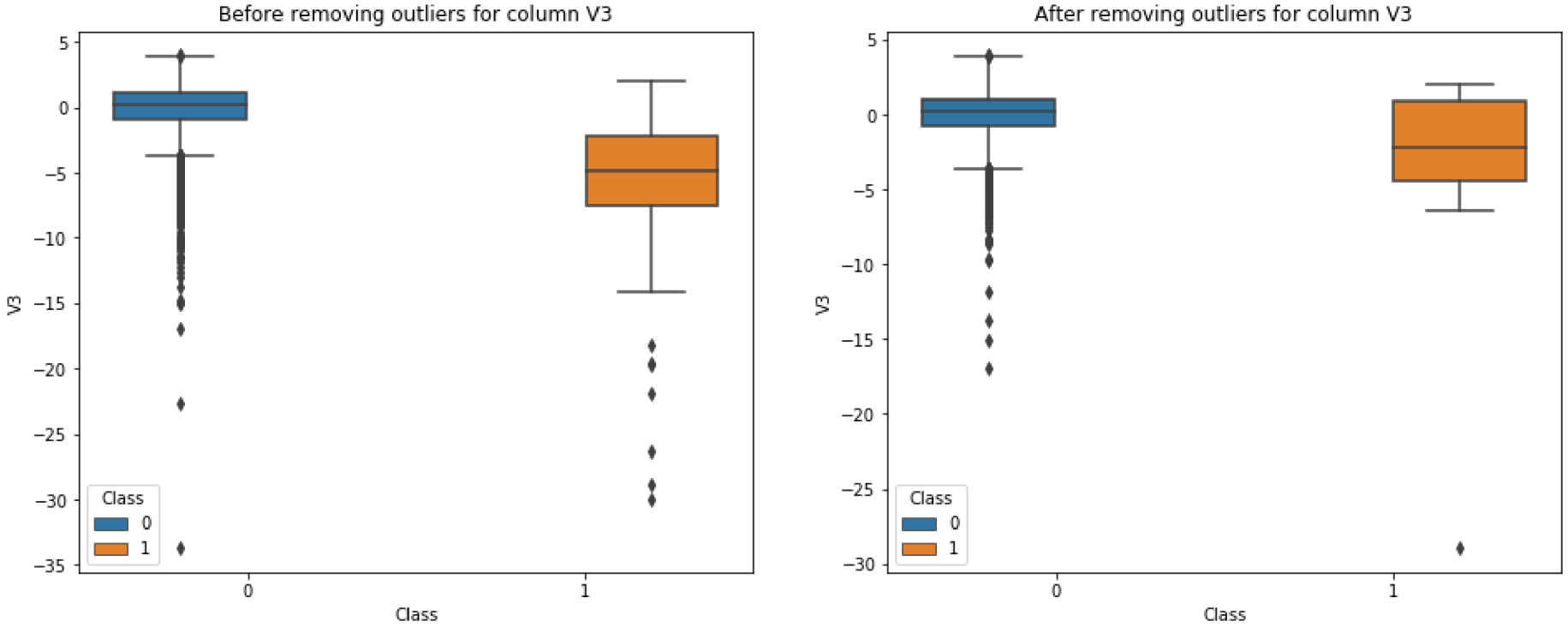
In [50]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V1")
ax = sns.boxplot(x="Class", y = "V1", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V1")
ax = sns.boxplot(x="Class", y = "V1", data= FinalData, hue = "Class")
```



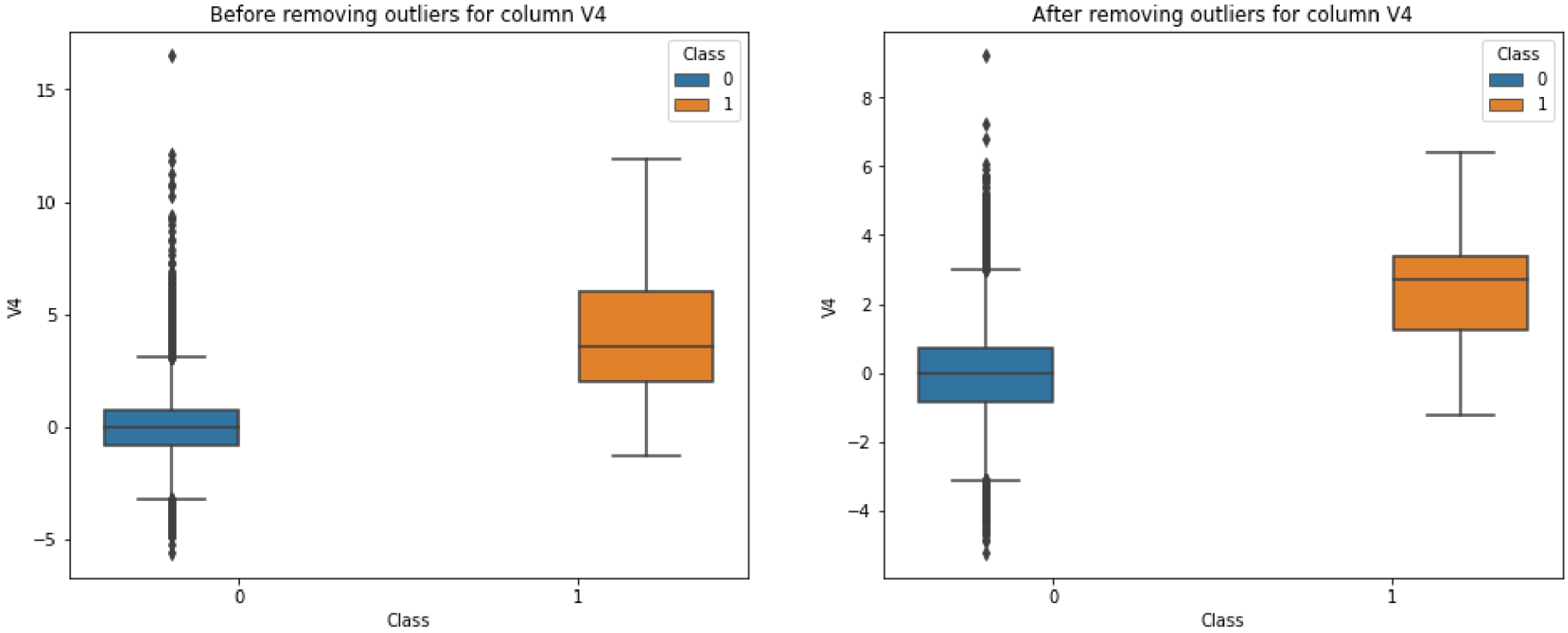
In [51]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V2")
ax = sns.boxplot(x="Class", y = "V2", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V2")
ax = sns.boxplot(x="Class", y = "V2", data= FinalData, hue = "Class")
```



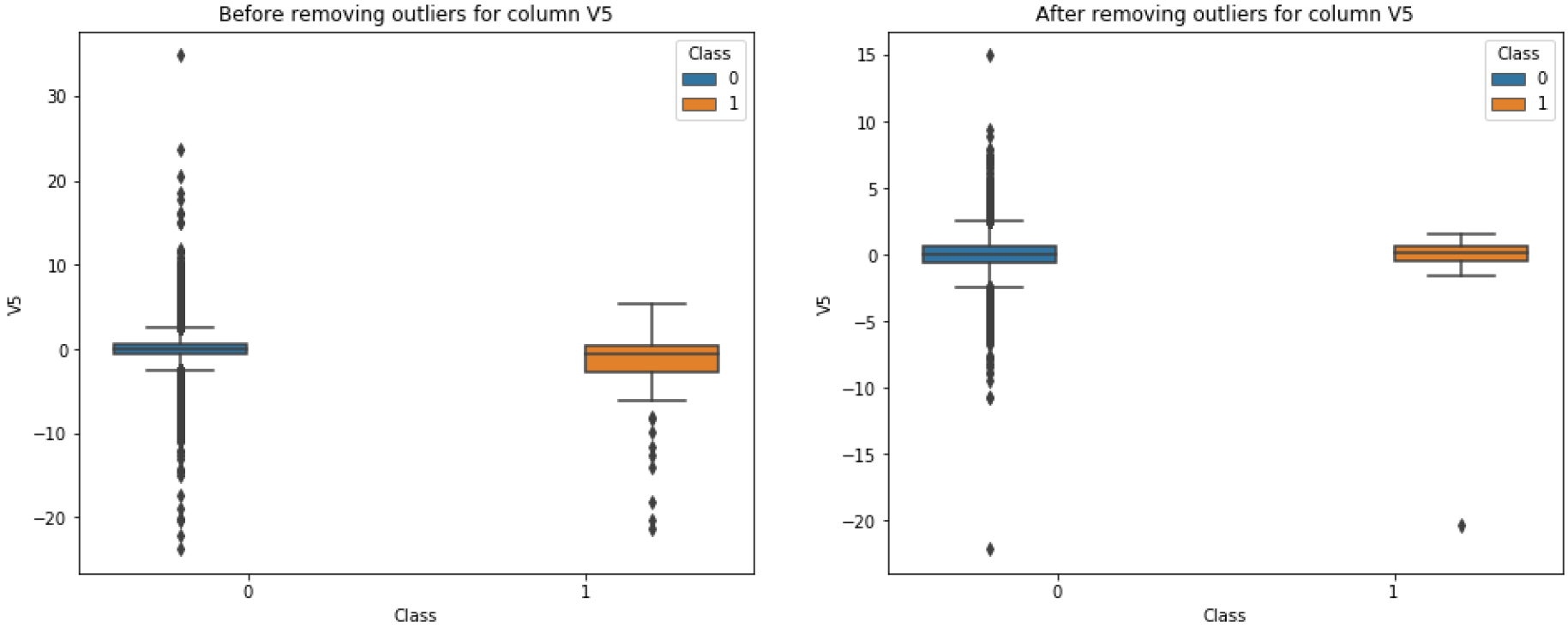
In [52]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V3")
ax = sns.boxplot(x="Class", y = "V3", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V3")
ax = sns.boxplot(x="Class", y = "V3", data= FinalData, hue = "Class")
```



In [53]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V4")
ax = sns.boxplot(x="Class", y = "V4", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V4")
ax = sns.boxplot(x="Class", y = "V4", data= FinalData, hue = "Class")
```



In [54]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V5")
ax = sns.boxplot(x="Class", y = "V5", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V5")
ax = sns.boxplot(x="Class", y = "V5", data= FinalData, hue = "Class")
```



It can easily be observed in all of the above boxplots corresponding to columns V1, V2, V3, V4, V5, that most of the outliers for points belongs to class 1 has been removed. Furthermore, many of the outliers for points belong to class 0 has also been removed. For example for class 0 in plot 1 for column V1 all of the outliers less than -35 have been removed. Similarly, for class 0 in plot 3 for column V3 all the outliers less than -18 have been removed. Similarly, for class 0 in plot 5 for column V5 all of the outliers greater than 15 have been removed.

Detecting outliers for 'k' value 5

We have assumed that 50% of total points in our data set are outliers.

In [55]:
```
newData = pd.read_csv("NewCreditCard.csv")

newData.head()
```

Out[55]:

	Unnamed: 0	Time	V1	V2	V3	V4	V5	V6	V7	V8	...	V21	V22	V23	
0	74072	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	...	0.018695	0.357714	-0.241636	-0
1	37724	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	...	0.513156	1.328134	-0.465055	-0
2	207494	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	...	0.253490	0.800281	-0.097207	0
3	180930	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	...	-0.145317	0.581410	-0.479921	-0
4	271725	164713.0	-1.424896	1.073129	0.105897	-0.941510	0.437835	0.584503	0.103029	1.050873	...	0.094488	0.238311	-0.060676	-0

5 rows × 32 columns

In [56]:
```
FinalData = newData.drop("Unnamed: 0", axis = 1)

FinalData.head()
```

Out[56]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	55413.0	1.010050	-0.476833	1.826221	2.930597	-1.174913	1.425051	-1.300402	0.581938	1.446847	...	0.018695	0.357714	-0.241636	-0.
1	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	-1.197673	...	0.513156	1.328134	-0.465055	-0.
2	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	0.405572	...	0.253490	0.800281	-0.097207	0.
3	124753.0	-1.215556	1.259983	3.236210	4.503750	-0.472941	2.304407	-0.756308	0.550394	0.525018	...	-0.145317	0.581410	-0.479921	-0.
4	164713.0	-1.424896	1.073129	0.105897	-0.941510	0.437835	0.584503	0.103029	1.050873	-0.247711	...	0.094488	0.238311	-0.060676	-0.

5 rows × 31 columns

In [57]:
```
FinalData.shape
```

Out[57]:
```
(50000, 31)
```

In [58]:
```
lof = LocalOutlierFactor(n_neighbors=5, algorithm='auto', metric='minkowski', p=2, metric_params=None, contamination=0.5,

outlierArray = lof.fit_predict(FinalData)

outlierArray
```

Out[58]:
```
array([-1,  1,  1, ..., -1, -1,  1])
```

Here, we got an array, where row corresponding to array element 1 in our dataset is an inlier and row corresponding to array element to -1 in our dataset is an oulier

In [59]:
```
len(outlierArray)
```

Out[59]:
```
50000
```

Calculating total number of outlier and inliers

In [60]:
```
countOutliers = 0
countInliers = 0
for i in range(50000):
    if outlierArray[i] == -1:
        countOutliers += 1
    else:
        countInliers += 1
print("Total number of outliers = "+str(countOutliers))
print("Total number of inliers = "+str(countInliers))
```

```
Total number of outliers = 25000
Total number of inliers = 25000
```

In [61]:
```
FinalData2 = FinalData.copy()
```

In [62]:
```
FinalData2.shape
```

Out[62]:
```
(50000, 31)
```

Removing Outliers

In [63]:
```
for i in range(50000):
    if outlierArray[i] == -1:
        FinalData.drop(i, inplace = True)
FinalData.head()
```

Out[63]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
1	39079.0	-1.233197	-0.282540	1.665411	-1.940511	-0.224044	0.291964	-0.678048	0.484719	-1.197673	...	0.513156	1.328134	-0.465055	-0
2	136702.0	1.917463	0.068259	-1.753261	0.585406	0.183106	-1.477567	0.450019	-0.481973	0.405572	...	0.253490	0.800281	-0.097207	0
5	145706.0	-0.011162	0.750724	0.337352	-0.669074	0.354770	-0.880356	0.847686	-0.018653	-0.184605	...	-0.211756	-0.509787	0.079205	0
6	154725.0	1.982318	-0.535051	-0.214848	0.521755	-0.983215	-0.640933	-0.727199	0.003512	1.439345	...	0.149974	0.598656	0.178110	0
7	32724.0	-0.455484	0.670802	1.465590	-1.228577	-0.004430	-0.935596	0.801324	-0.214961	0.259080	...	-0.066009	-0.013992	-0.025550	0

5 rows × 31 columns

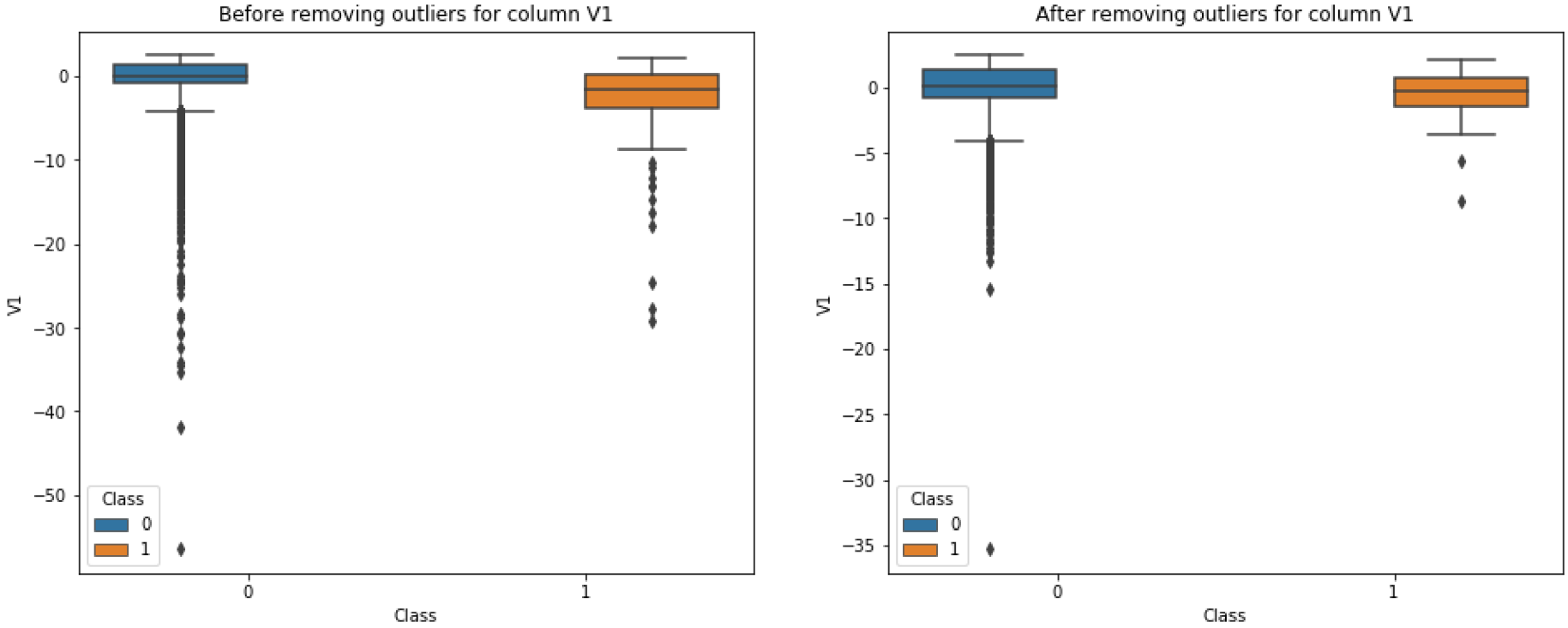
In [64]:
```
FinalData2.shape
```

Out[64]:
```
(50000, 31)
```

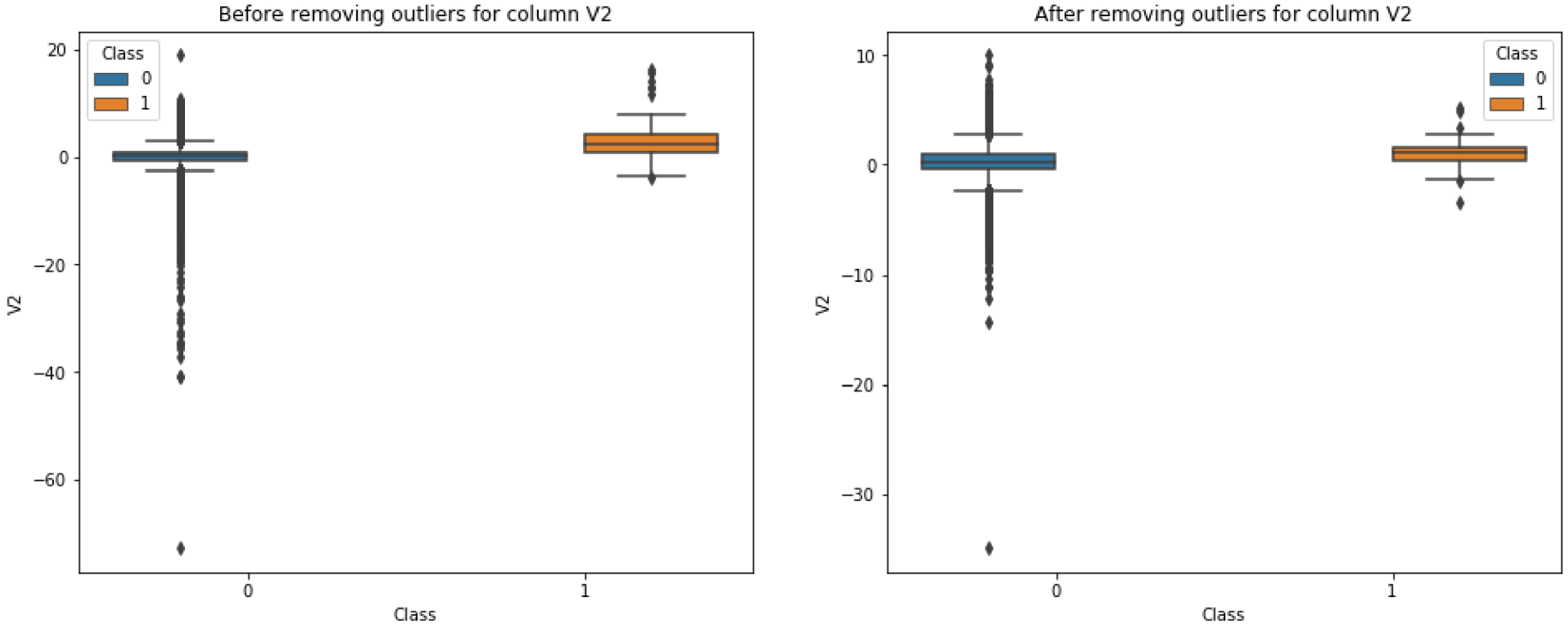
In [65]:
```
FinalData.shape
```

Out[65]:
```
(25000, 31)
```

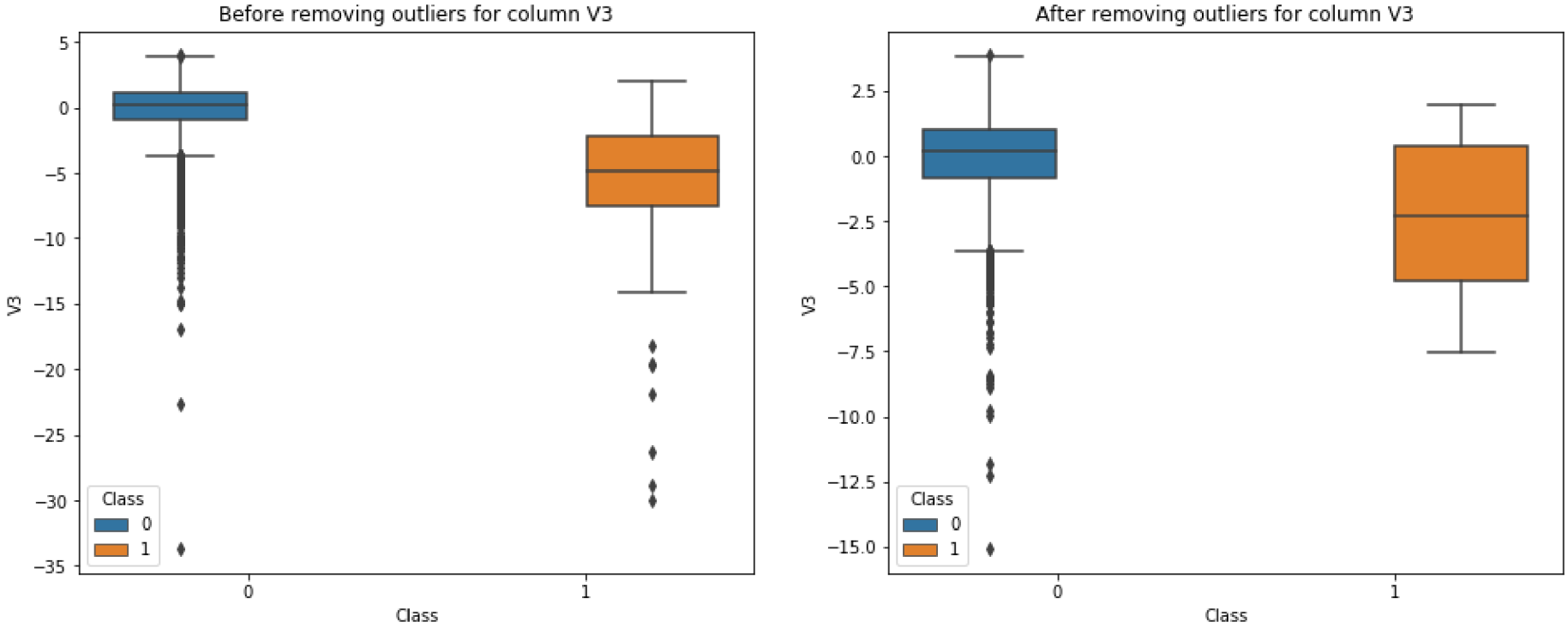
In [66]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V1")
ax = sns.boxplot(x="Class", y = "V1", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V1")
ax = sns.boxplot(x="Class", y = "V1", data= FinalData, hue = "Class")
```



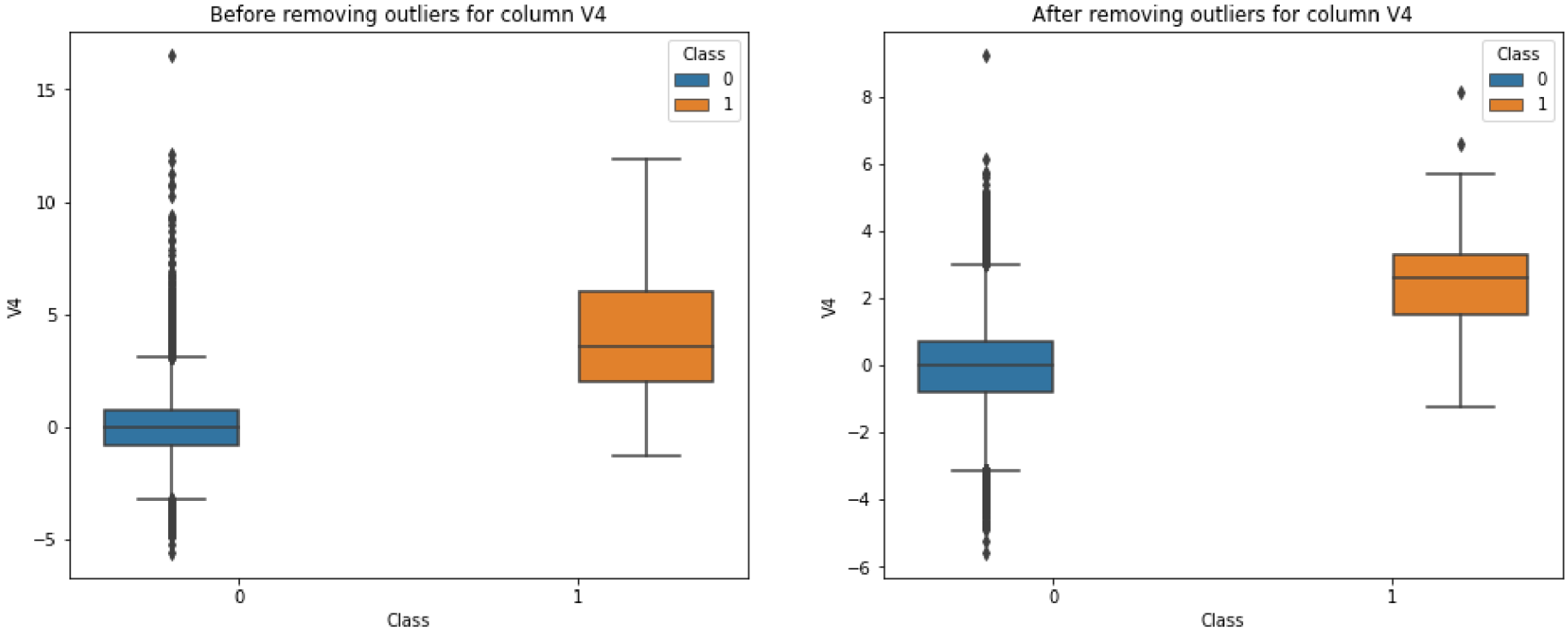
In [67]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V2")
ax = sns.boxplot(x="Class", y = "V2", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V2")
ax = sns.boxplot(x="Class", y = "V2", data= FinalData, hue = "Class")
```



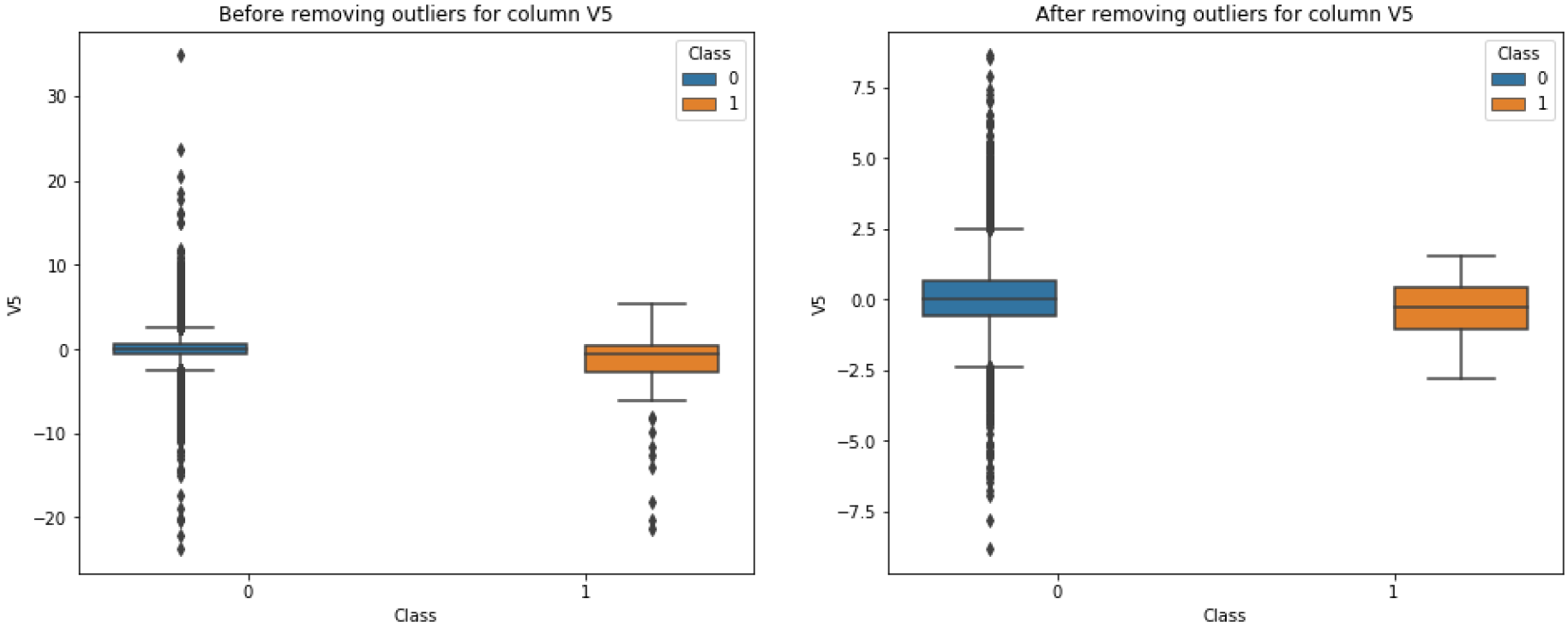
In [68]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V3")
ax = sns.boxplot(x="Class", y = "V3", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V3")
ax = sns.boxplot(x="Class", y = "V3", data= FinalData, hue = "Class")
```



In [69]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V4")
ax = sns.boxplot(x="Class", y = "V4", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V4")
ax = sns.boxplot(x="Class", y = "V4", data= FinalData, hue = "Class")
```



In [70]:
```
fig = plt.figure(figsize = (16,6))

plt.subplot(1, 2, 1)
plt.title("Before removing outliers for column V5")
ax = sns.boxplot(x="Class", y = "V5", data= FinalData2, hue = "Class")

plt.subplot(1, 2, 2)
plt.title("After removing outliers for column V5")
ax = sns.boxplot(x="Class", y = "V5", data= FinalData, hue = "Class")
```



It can easily be observed in all of the above boxplots corresponding to columns V1, V2, V3, V4, V5, that most of the outliers for points belongs to class 1 has been removed. Furthermore, many of the outliers for points belong to class 0 has also been removed. For example for class 0 in plot 1 for column V1 all of the outliers less than -17 have been removed. Similarly, for class 0 in plot 3 for column V3 all the outliers less than -15 have been removed. Similarly, for class 0 in plot 5 for column V5 all of the outliers greater than 8.5 have been removed.

In conclusion, for k = 5 more outliers have been removed as compared to k = 2 or 3. Therefore, k = 5 is the best value for number of neighbors.