In [ ]:

```
#Ref-"https://spd.group/machine-learning/credit-card-fraud-detection/#:~:text=obvious%20fraud%20activities-,What%20is%20Credit%20Card%20Fraud%20Detection%3F,ways%20and%20in%20many%20industries."

#  In recent years credit card fraud has become one of the growing problem.
#A large financial loss has greatly affected individual person using
#credit card and also the merchants and banks.
# What are  Credit Card Fraud -  act committed by any person who, with intent to defraud, uses a credit card
#                                that has been revoked, cancelled, reported lost, or stolen to obtain anything of value.
# What is the difference between ML Credit Card Fraud Detection and Conventional Fraud Detection?
# Machine Learning-based Fraud Detection:
  #Detecting fraud automatically
  #Real-time streaming
  #Less time needed for verification methods
  #Identifying hidden correlations in data
  #Conventional Fraud Detection:

# To Know More Please Refer The Above Link
```

In [ ]:

```
# MatplotLib- Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python
# Matplot Library Ref - "https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/#:~:text=Matplotlib%20is%20a%20cross%2Dplatform,embed%20plots%20in%20GUI%20applications."

# Pandas - Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks.
#It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.
# Pandas Library Ref - "https://www.journaldev.com/29055/python-pandas-module-tutorial"

# Numpy - NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for
#processing those arrays.
#Using NumPy, mathematical and logical operations on arrays can be performed.
# Numpy Ref- " https://www.mygreatlearning.com/blog/python-numpy-tutorial/#:~:text=NumPy%2C%20which%20stands%20for%20Numerical,stands%20for%20'Numerical%20Python'."
```

In [ ]:

```
# K-Means Clustering

# Importing the libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:

```
# Dataset - Anonymized credit card transactions labeled as fraudulent or genuine
# Source - https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
#The dataset 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.
# To Know More About the dataset.Please Check The link-"https://machinelearningmastery.com/imbalanced-classification-with-the-fraudulent-credit-card-transactions-dataset/"
```

In [ ]:

```
dataset = pd.read_csv('/content/creditcard.csv')
dataset
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V2
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.27783
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	0.63867
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.77167
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.00527
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.79827
...	...	...	...	...	...	...	...	...	...	...	...	...	
19968	30686	-1.116462	1.329355	-0.853334	0.905279	-0.112579	-0.705548	0.401447	0.659747	-1.028598	...	0.214942	0.64954
19969	30688	-1.803451	1.319525	0.793831	0.447426	0.191135	-0.275629	0.946229	-0.573614	0.861838	...	-0.005811	0.37034
19970	30688	1.263475	-1.007268	1.053241	-0.518287	-1.615187	0.108571	-1.270099	0.271352	-0.071871	...	-0.452567	0.78831
19971	30688	-0.922380	0.559681	1.969629	0.550002	0.599025	0.005809	0.999657	-0.310841	0.013936	...	-0.142477	0.12975
19972	30690	-0.763278	0.785587	2.589429	-0.228285	-0.068595	0.136331	0.413511	0.131367	-0.387970	...	-0.038228	0.11657

19973 rows × 31 columns

In [ ]:

```
# iloc()-integer-location based indexing for selection by position.iloc[] is primarily in
teger position based (from 0 to length-1 of the axis), but may also be used with a boolea
n array.
# Ref-" https://www.askpython.com/python/built-in-methods/python-iloc-function"
# First parameter inside the iloc()- rows
# Second Parameter inside the iloc()- column // Randomly Chosen
```

In [16]:

```
X = dataset.iloc[:,[3,15]].values
X
```

Out[16]:

```
array([[ 2.53634674,  1.46817697],
       [ 0.16648011,  0.63555809],
       [ 1.77320934,  2.34586495],
       ...,
       [ 1.0532413 , -0.43025556],
       [ 1.9696292 , -0.87943768],
       [ 2.58942904,  0.74155663]])
```

In [17]:

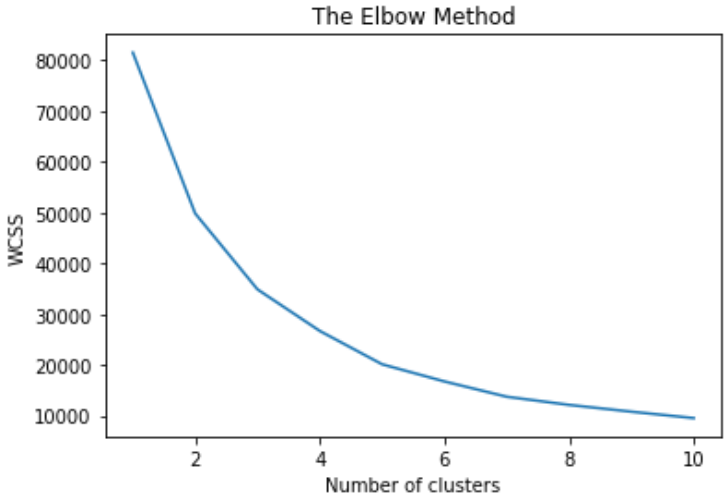
```
# Using the elbow method to find the optimal number of clusters
#The WCSS ( or Within Cluster Sum of Squares ) was caluated and plotted to find the optim
al number of clusters. The "Elbow Method" was used to find the optimal number of clusters
.

#Once the optimal number of clusters were found the model was reinitalised with the n_clu
```

```
ster arguments begin passed with the optimal number of clusters found using the "Elbow Me
thod".

from sklearn.cluster import KMeans
wcss =[]
for i in range (1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter =300, n_init = 10, rand
om_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plot the graph to visualize the Elbow Method to find the optimal number of cluster
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [20]:

```
# Kmeans Ref-"https://scikit-learn.org/stable/modules/clustering.html#k-means"
        # Reference 2 - " https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-mea
ns.html"


#  KMean Syntax Reference-"https://github.com/scikit-learn/scikit-learn/blob/baf828ca1/sk
learn/cluster/_kmeans.py#L763"
# Applying KMeans to the dataset with the optimal number of cluster

kmeans=KMeans(n_clusters= 3, init = 'k-means++', max_iter = 300, n_init = 10, random_sta
te = 0)
y_kmeans = kmeans.fit_predict(X)
```

In [ ]:

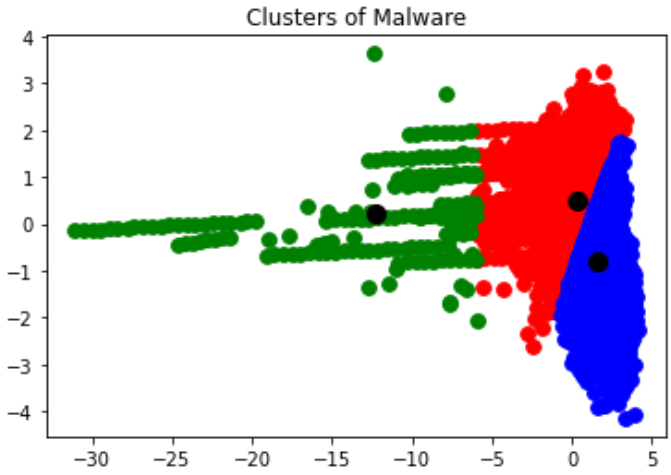
```
# Insights from the clusters KMeans -
```

In [22]:

```
# Visualising the clusters

plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 60, c = 'red', label = 'Cluste
r1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 60, c = 'blue', label = 'Clust
er2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 60, c = 'green', label = 'Clus
ter3')
#plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 60, c = 'violet', label = 'Clu
ster4')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = '
black', label = 'Centroids')
plt.title('Clusters of Malware')

plt.show()
```



In [ ]:

```
#Accuracy metrics
#As opposed to classfication, it is difficult to assess the quality of results from clust
ering.
#Here, a metric cannot depend on the labels but only on the goodness of split.
#Secondly, we do not usually have true labels of the observations when we use clustering.

#There are internal and external goodness metrics. External metrics use the information a
bout the known true split while internal metrics-
# do not use any external information and assess the goodness of clusters based only on t
he initial data. The optimal number of clusters -
#is usually defined with respect to some internal metrics.

##External Goodness Metrics

#F-measure, Normalized Mutual Information(the average mutual information between every pa
ir of clusters and their class), Rand Index etc.

##Internal Goodness Metrics

#Davies-Bouldin index,Silhouette index,Dunn index,Partition Coefficient, Entropy,Separati
on Index,Xie and Beni's Index etc.

##Normalized Mutual Information (NMI)

#Mutual Information of two random variables is a measure of the mutual dependence between
the two variables.
#Normalized Mutual Information is a normalization of the Mutual Information (MI) score to
scale the results between 0
# (no mutual information) and 1 (perfect correlation). In other words, 0 means dissimilar
and 1 means a perfect match.

##Adjusted Rand Score (ARS)

#Adjusted Rand Score on the other hand, computes a similarity measure between two cluster
s.
#ARS considers all pairs of samples and counts pairs that are assigned in the same or dif
ferent clusters in the predicted and true clusters.
#0 is the lowest similarity and 1 is the highest.

##Reffered From -"https://www.kaggle.com/code/vipulgandhi/kmeans-detailed-explanation/not
ebook"
```

In [26]:

```
labels = kmeans.labels_

# check how many of the samples were correctly labeled

correct_labels = sum(y_kmeans == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y_kmeans.
size))
```

```
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y_kmeans.size)))
```

```
Result: 19973 out of 19973 samples were correctly labeled.
Accuracy score: 1.00
```