# NAME: AVIREDDY NVSRK ROHAN

# REG.NO: 19BCE1180

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from matplotlib import pyplot as plt
```

In [2]:

```
import matplotlib.pyplot as plt
```

In [3]:

```
from sklearn.cluster import KMeans
```

In [4]:

```
from matplotlib import cm
```

# KDD 99 DATA

In [5]:

```
data=pd.read_csv("/content/drive/MyDrive/ML_LAB/corrected",header=None)
data.columns=['duration','protocol_type','service','flag','src_bytes','dst_bytes'
```

In [ ]:

```
data.head()
```

Out[6]:

| | duration | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragment | urgent | hot |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | udp | private | SF | 105 | 146 | 0 | 0 | 0 | 0 |
| **1** | 0 | udp | private | SF | 105 | 146 | 0 | 0 | 0 | 0 |
| **2** | 0 | udp | private | SF | 105 | 146 | 0 | 0 | 0 | 0 |
| **3** | 0 | udp | private | SF | 105 | 146 | 0 | 0 | 0 | 0 |
| **4** | 0 | udp | private | SF | 105 | 146 | 0 | 0 | 0 | 0 |

In [6]:

```
le=LabelEncoder()
data["protocol_type"]=le.fit_transform(data["protocol_type"])
data["service"]=le.fit_transform(data["service"])
data["flag"]=le.fit_transform(data["flag"])
data["protocol_type"]=data["protocol_type"].astype("object")
data["service"]=data["service"].astype("object")
data["flag"]=data["flag"].astype("object")
data["is_host_login"]=data["is_host_login"].astype("object")
data["is_guest_login"]=data["is_guest_login"].astype("object")
data["logged_in"]=data["logged_in"].astype("object")
```

In [7]:

```
cond=((data['class']=="normal.") | (data['class']=="neptune.") | (data['class']==
```

In [8]:

```
new_data=data[cond]
```

In [9]:

```
multi_label=new_data["class"]
new_data.drop(["class"],inplace=True,axis=1)
multi_label=le.fit_transform(multi_label)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4174: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  errors=errors,
```

In [10]:

```
st_obj=StandardScaler()
data_std=st_obj.fit_transform(new_data)
data_std=pd.DataFrame(data_std)
```

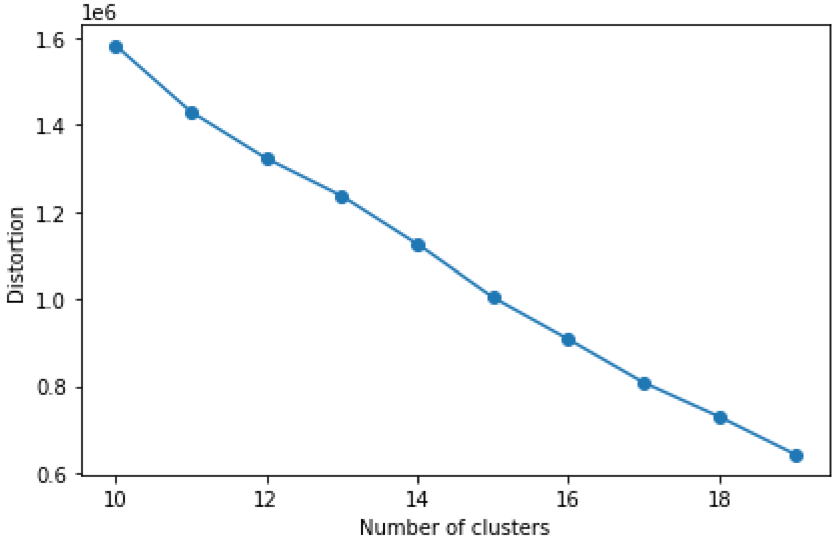
# KMEANS MODEL

In [27]:

```
km1 = KMeans(n_clusters=3,
            init='random',
            n_init=1,
            max_iter=2,
            tol=1e-04,
            random_state=0)
y_km1 = km1.fit_predict(data_std)
```

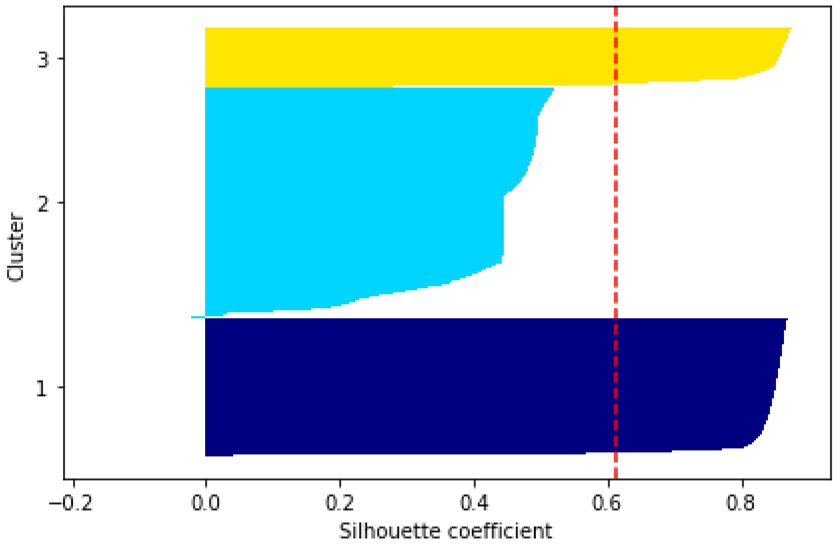
In [24]:

```
distortions = []
for i in range(10, 20):
    km2 = KMeans(n_clusters=i,
                init='k-means++',
                n_init=10,
                max_iter=300,
                random_state=0)
    km2.fit(data_std)
    distortions.append(km2.inertia_)
plt.plot(range(10, 20), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.tight_layout()
#plt.savefig('./figures/elbow.png', dpi=300)
plt.show()
```



In [ ]:

```
from sklearn.metrics import silhouette_samples
km = KMeans(n_clusters=3,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(data_std)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(data_std, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
# plt.savefig('./figures/silhouette.png', dpi=300)
plt.show()
```

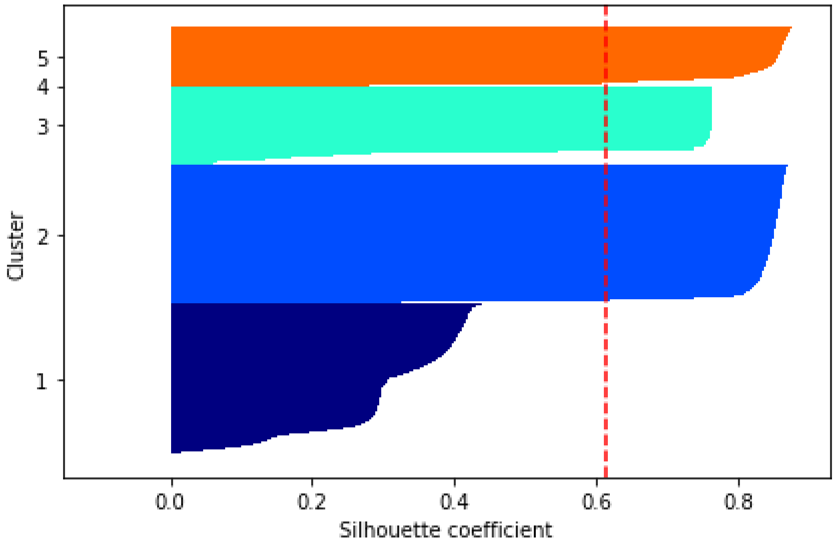


**inference: 2nd cluster is not efficient because it is below the slc**

### threshold

In [30]:

```
from sklearn.metrics import silhouette_samples
km = KMeans(n_clusters=5,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(data_std)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(data_std, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
# plt.savefig('./figures/silhouette.png', dpi=300)
plt.show()
```



## increasing the number of clusters gave a better clustering for most of points but cluster 1 is still less than the sc threshold(silhouette coeff)

## country data from kaggle: [https://www.kaggle.com/rohan0301/unsupervised-learning-on-country-data?select=data-dictionary.csv (https://www.kaggle.com/rohan0301/unsupervised-learning-on-country-data?select=data-dictionary.csv)](https://www.kaggle.com/rohan0301/unsupervised-learning-on-country-data?select=data-dictionary.csv)

In [32]:
```
data2=pd.read_csv("/content/drive/MyDrive/ML_LAB/kmeans data.zip (Unzipped Files)
```

In [ ]:
```
data2.head()
```

Out[20]:

| | country | child_mort | exports | health | imports | income | inflation | life_expec | total_fer | gdpp |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 90.2 | 10.0 | 7.58 | 44.9 | 1610 | 9.44 | 56.2 | 5.82 | 553 |
| **1** | Albania | 16.6 | 28.0 | 6.55 | 48.6 | 9930 | 4.49 | 76.3 | 1.65 | 4090 |
| **2** | Algeria | 27.3 | 38.4 | 4.17 | 31.4 | 12900 | 16.10 | 76.5 | 2.89 | 4460 |
| **3** | Angola | 119.0 | 62.3 | 2.85 | 42.9 | 5900 | 22.40 | 60.1 | 6.16 | 3530 |
| **4** | Antigua and Barbuda | 10.3 | 45.5 | 6.03 | 58.9 | 19100 | 1.44 | 76.8 | 2.13 | 12200 |

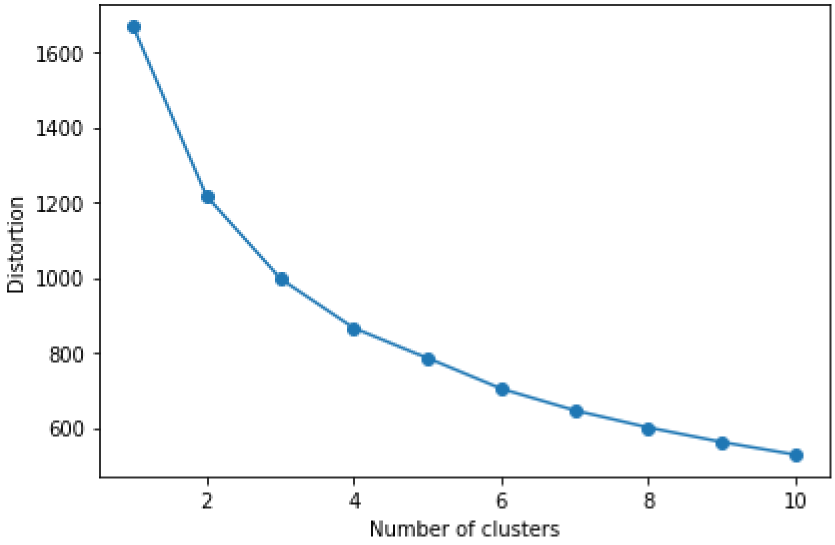
In [33]:
```
data2["country"]=le.fit_transform(data2["country"])
```

In [34]:
```
data_std2=st_obj.fit_transform(data2)
data_std2=pd.DataFrame(data_std2)
```

In [35]:
```
km2 = KMeans(n_clusters=3,
            init='random',
            n_init=1,
            max_iter=2,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(data_std2)
```

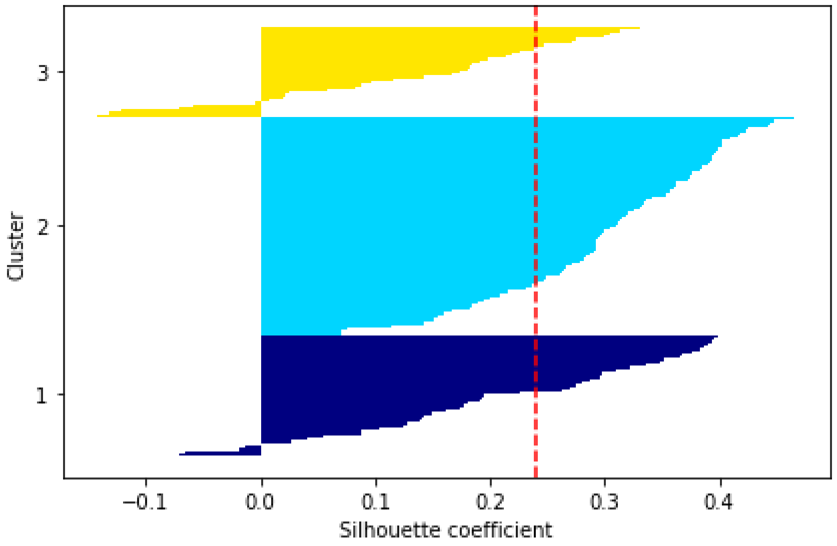
In [36]:

```
distortions = []
for i in range(1, 11):
    km2 = KMeans(n_clusters=i,
                init='k-means++',
                n_init=10,
                max_iter=300,
                random_state=0)
    km2.fit(data_std2)
    distortions.append(km2.inertia_)
plt.plot(range(1, 11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.tight_layout()
#plt.savefig('./figures/elbow.png', dpi=300)
plt.show()
```



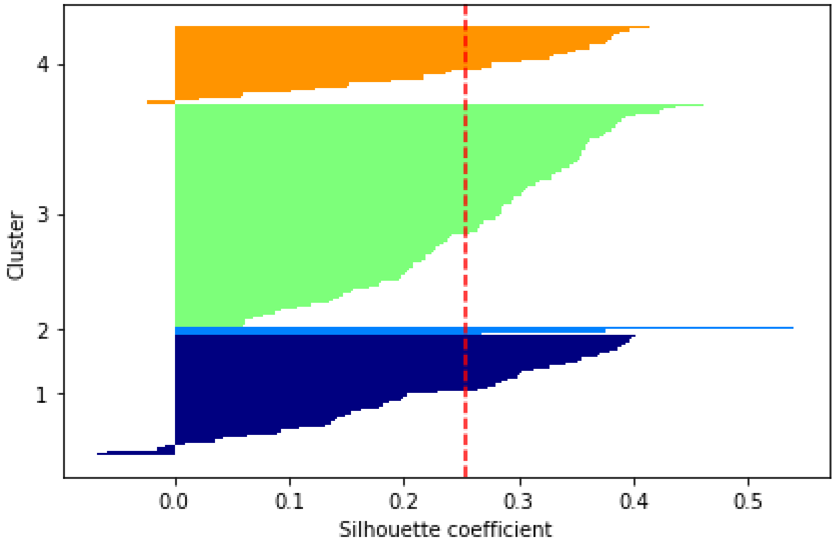
In [ ]:

```
from sklearn.metrics import silhouette_samples
km = KMeans(n_clusters=3,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(data_std2)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(data_std2, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
# plt.savefig('./figures/silhouette.png', dpi=300)
plt.show()
```



In [38]:

```
from sklearn.metrics import silhouette_samples
km2 = KMeans(n_clusters=4,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km2.fit_predict(data_std2)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(data_std2, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')

plt.tight_layout()
# plt.savefig('./figures/silhouette.png', dpi=300)
plt.show()
```



**4 and greater than 4 k value gives better clustering**

**because all clusters cross the sc threshold**