Clustering the NASA Exoplanet Archive With K-means to Find Earth-II Candidates

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_samples, silhouette_score
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.cm as cm
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
import warnings
warnings.filterwarnings('ignore')
```

EDA | Feature Engineering

In [2]:
```
# load nasa exoplanet archive dataset #
# https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=PS #
# confirmed planets -> planetary systems #
df = pd.read_excel('/Users/shaner/Desktop/nasa_exoplanets.xlsx')
df.head()
```

Out[2]:

	pl_name	hostname	pl_orbper	pl_orbpererr1	pl_orbpererr2	pl_orbperlim	pl_orbsmax	pl_orbsmaxerr1	pl_orbsmaxerr2	pl_orbsmaxlim	...	pl_masseerr2	pl_masselim	pl_orbeccen	pl_orbeccenerr1
0	11 Com b	11 Com	NaN	NaN	NaN	NaN	1.21	0.06	-0.05	0.0	...	NaN	NaN	NaN	NaN
1	11 Com b	11 Com	326.03000	0.32	-0.32	0.0	1.29	0.05	-0.05	0.0	...	NaN	NaN	0.231	0.005
2	11 UMi b	11 UMi	NaN	NaN	NaN	NaN	1.51	0.06	-0.05	0.0	...	NaN	NaN	NaN	NaN
3	11 UMi b	11 UMi	516.21997	3.20	-3.20	0.0	1.53	0.07	-0.07	0.0	...	NaN	NaN	0.080	0.030
4	11 UMi b	11 UMi	516.22000	3.25	-3.25	0.0	1.54	0.07	-0.07	0.0	...	NaN	NaN	0.080	0.030

5 rows × 26 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34979 entries, 0 to 34978
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   pl_name          34979 non-null  object
 1   hostname         34979 non-null  object
 2   pl_orbper        31918 non-null  float64
 3   pl_orbpererr1    30584 non-null  float64
 4   pl_orbpererr2    30583 non-null  float64
 5   pl_orbperlim     31918 non-null  float64
 6   pl_orbsmax       19324 non-null  float64
 7   pl_orbsmaxerr1   5031 non-null   float64
 8   pl_orbsmaxerr2   5030 non-null   float64
 9   pl_orbsmaxlim    22103 non-null  float64
 10  pl_rade          24108 non-null  float64
 11  pl_radeerr1      23387 non-null  float64
 12  pl_radeerr2      23387 non-null  float64
 13  pl_radelim       26853 non-null  float64
 14  pl_masse         3515 non-null   float64
 15  pl_masseerr1     3264 non-null   float64
 16  pl_masseerr2     3264 non-null   float64
 17  pl_masselim      3547 non-null   float64
 18  pl_orbeccen      17533 non-null  float64
 19  pl_orbeccenerr1  3011 non-null   float64
 20  pl_orbeccenerr2  3010 non-null   float64
 21  pl_orbeccenlim   20279 non-null  float64
 22  pl_eqt           16029 non-null  float64
 23  pl_eqterr1       1793 non-null   float64
 24  pl_eqterr2       1793 non-null   float64
 25  pl_eqtlim        18774 non-null  float64
dtypes: float64(24), object(2)
memory usage: 6.9+ MB
```

In [4]:
```
# drop rows with null values #
df.dropna(inplace=True)

# keep only relevant columns #
columns_to_keep = ['pl_name','pl_rade','pl_masse','pl_orbeccen','pl_eqt']
df = df[columns_to_keep]

df.head()
```

Out[4]:

	pl_name	pl_rade	pl_masse	pl_orbeccen	pl_eqt
103	55 Cnc e	2.080	7.8100	0.061	1958.0
205	CoRoT-1 b	15.917	340.0781	0.071	1834.0
213	CoRoT-10 b	10.870	874.0000	0.530	600.0
222	CoRoT-12 b	16.140	291.4380	0.070	1442.0
248	CoRoT-19 b	14.460	352.7800	0.047	2000.0

In [5]:
```
# new row of data for earth #
new_row = {'pl_name': 'Earth','pl_rade': 1,'pl_masse': 1,'pl_orbeccen': 0.0167,'pl_eqt': 255}

# add the new row using the loc indexer #
df.loc[len(df)] = new_row
```

In [6]:
```
# remove whitespace and case-sensitivity to ensure #
# duplicates are dropped #
df['pl_name'] = df['pl_name'].str.strip()
df['pl_name'] = df['pl_name'].str.lower()
df.drop_duplicates(subset='pl_name', keep='last', inplace=True)
```

In [7]:
```
df.head(20)
```

Out[7]:

	pl_name	pl_rade	pl_masse	pl_orbeccen	pl_eqt
103	55 cnc e	2.080	7.81000	0.061	1958.0
205	corot-1 b	15.917	340.07810	0.071	1834.0
213	corot-10 b	10.870	874.00000	0.530	600.0
222	corot-12 b	16.140	291.43800	0.070	1442.0
248	corot-19 b	14.460	352.78000	0.047	2000.0
264	corot-20 b	9.420	1347.54000	0.562	1002.0
270	corot-22 b	4.880	12.20000	0.077	885.0
296	corot-4 b	13.340	228.83000	0.000	1074.0
301	corot-5 b	15.558	148.42000	0.090	1438.0
330	corot-9 b	11.949	266.97720	0.133	420.0
451	epic 220674823 c	2.500	5.80000	0.180	774.0
463	epic 249893012 b	1.950	8.75000	0.060	1616.0
464	epic 249893012 c	3.670	14.67000	0.070	990.0
465	epic 249893012 d	3.940	10.18000	0.150	752.0
511	gj 143 b	2.610	22.70000	0.188	422.0
571	gj 3470 b	3.880	13.73000	0.017	604.0
594	gj 367 b	0.699	0.63300	0.060	1365.0
632	gj 436 b	4.191	23.13802	0.150	686.0
824	hat-p-11 b	4.730	25.74300	0.198	878.0
846	hat-p-13 b	14.359	271.09800	0.021	1653.0

In [8]:
```
df.tail(20)
```

Out[8]:

	pl_name	pl_rade	pl_masse	pl_orbeccen	pl_eqt
33940	wasp-148 b	8.093	92.48853	0.2200	940.0
33958	wasp-150 b	11.994	2688.84180	0.3775	1460.0
34005	wasp-162 b	11.209	1652.71600	0.4340	910.0
34034	wasp-17 b	22.317	154.46538	0.0280	1771.0
34086	wasp-18 b	13.899	3241.84975	0.0051	2429.0
34104	wasp-185 b	14.011	311.47340	0.2400	1160.0
34123	wasp-19 b	15.861	366.77398	0.0126	2113.0
34134	wasp-193 b	16.410	44.17815	0.0560	1254.0
34167	wasp-22 b	12.554	177.98480	0.0230	1430.0
34261	wasp-34 b	13.675	187.51000	0.0380	1250.0
34424	wasp-5 b	12.184	502.15000	0.0380	1706.0
34457	wasp-54 b	18.528	202.13100	0.0670	1759.0
34493	wasp-59 b	8.687	274.27600	0.1000	670.0
34495	wasp-6 b	13.787	154.14755	0.0540	1184.0
34626	wasp-77 a b	13.787	529.81995	0.0074	1715.0
34655	wasp-80 b	11.198	170.99254	0.0020	825.0
34703	wasp-89 b	11.657	1875.19700	0.1930	1120.0
34785	wolf 503 b	2.043	6.26000	0.4100	790.0
34854	xo-7 b	15.390	225.34147	0.0380	1743.0
328	earth	1.000	1.00000	0.0167	255.0

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 303 entries, 103 to 328
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   pl_name      303 non-null    object
 1   pl_rade      303 non-null    float64
 2   pl_masse     303 non-null    float64
 3   pl_orbeccen  303 non-null    float64
 4   pl_eqt       303 non-null    float64
dtypes: float64(4), object(1)
memory usage: 14.2+ KB
```

Principal Component Analysis (PCA)

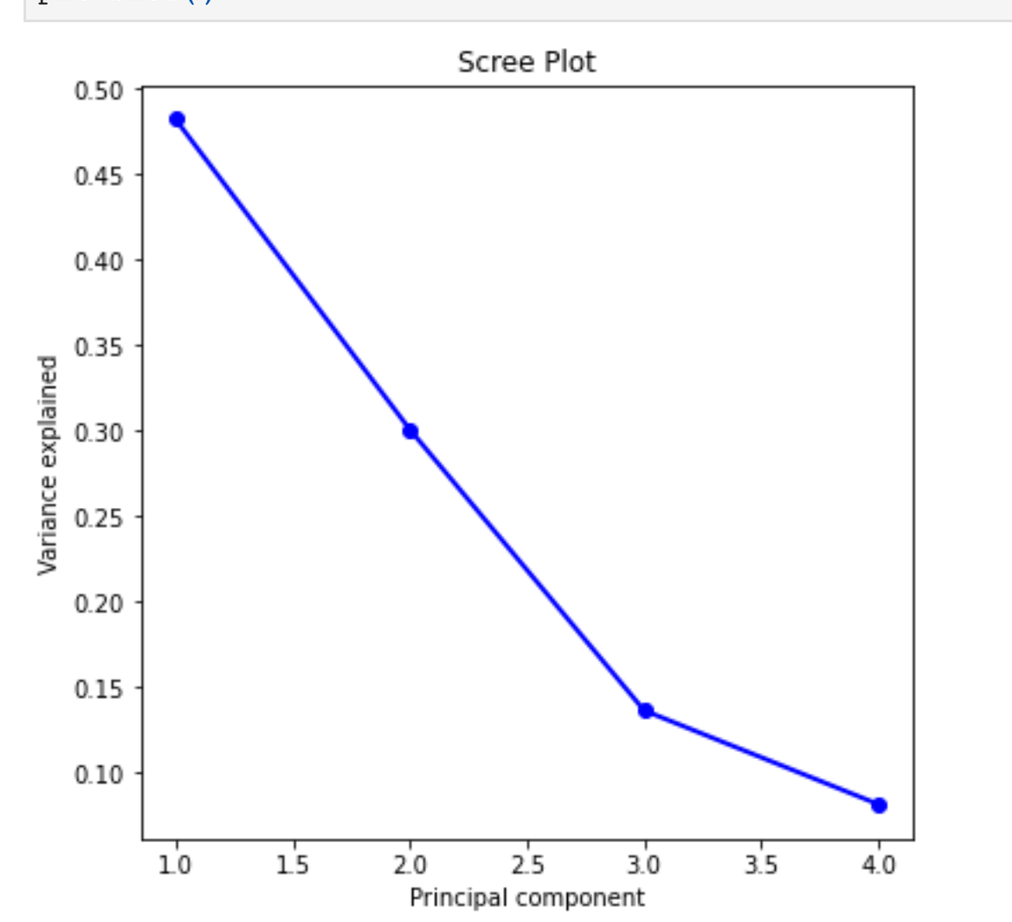
In [10]:
```
# select features for clustering and pca #
features = ['pl_rade','pl_masse','pl_orbeccen','pl_eqt']
```

In [11]:
```
# standardize the data #
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[features])
```

In [12]:
```
# perform pca #
pca = PCA(n_components=4)
pca_result = pca.fit_transform(scaled_data)
```

Scree Plot

In [13]:
```
# scree with matplotlib #
pc_values = np.arange(pca.n_components_) + 1

plt.figure(figsize=(6, 6))  # set the figure size to be square #

plt.plot(pc_values, pca.explained_variance_ratio_, marker='o', color='blue', linewidth=2)
plt.title('Scree Plot')
plt.xlabel('Principal component')
plt.ylabel('Variance explained')

plt.show()
```



In [14]:
```
print(pca.explained_variance_ratio_)
```

```
[0.48228785 0.30057672 0.1360567  0.08107874]
```

In [15]:
```
# calculate cumulative explained variance #
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratio_)

print(cumulative_explained_variance)
```

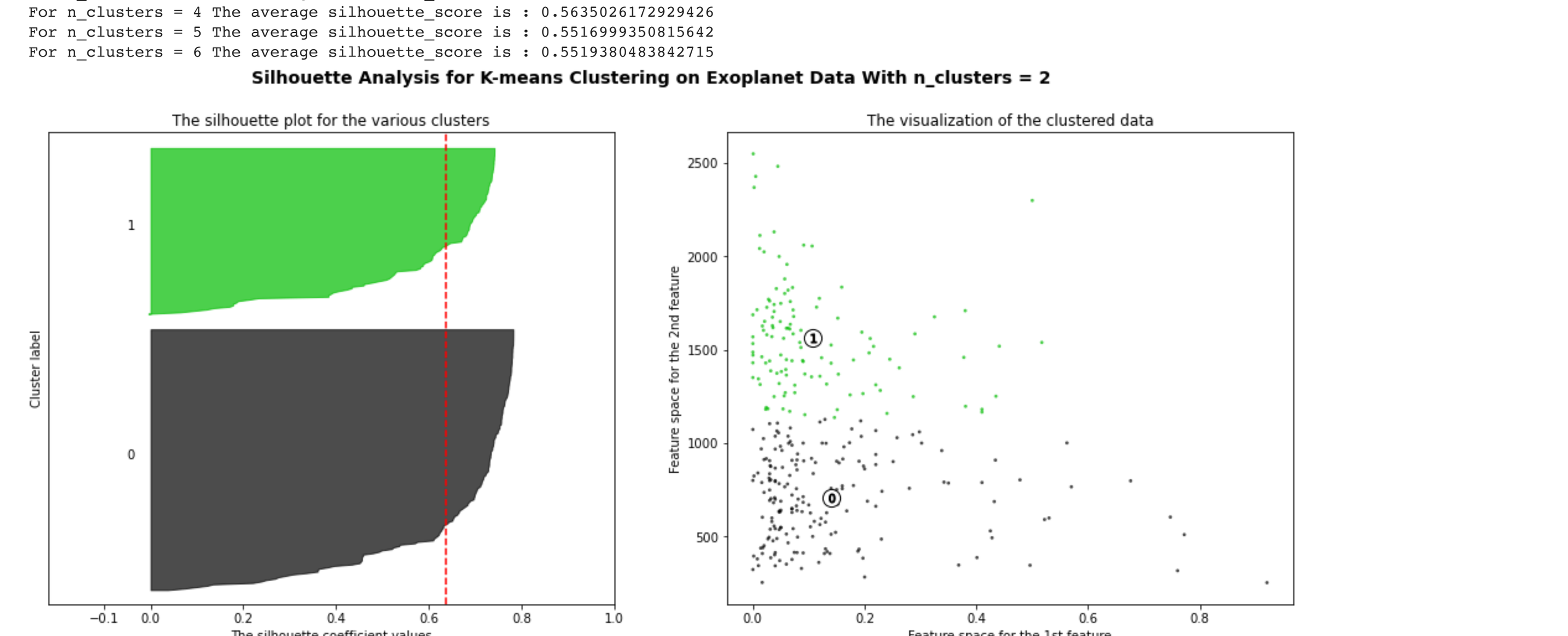
```
[0.48228785 0.78286456 0.91892126 1.        ]
```

Silhouette Analysis

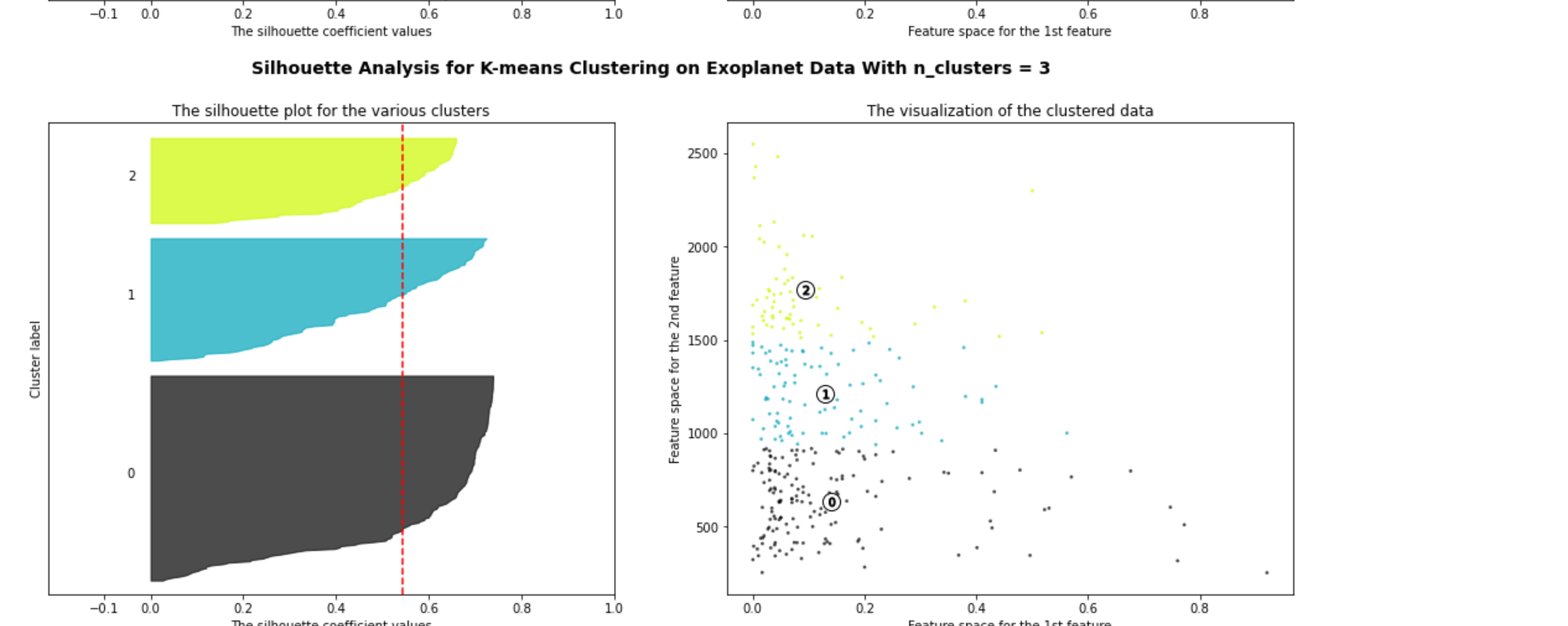
In [16]:
```
# silhouette analysis with matplotlib #
range_n_clusters = [2, 3, 4, 5, 6]

X = np.array(df)[:, -2:]

for n_clusters in range_n_clusters:
    # create a subplot with 1 row and 2 columns #
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # the 1st subplot is the silhouette plot #
    # the silhouette coefficient can range from -1, 1 but here all #
    # lie within [-0.22, 1] #
    ax1.set_xlim([-0.22, 1])
    # the (n_clusters+1)*10 is for inserting blank space between silhouette #
    # plots of individual clusters, to demarcate them clearly #
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # initialize the clusterer with n_clusters value and a random generator #
    # seed of 10 for reproducibility #
    clusterer = KMeans(n_clusters=n_clusters, n_init="auto", random_state=10)
    cluster_labels = clusterer.fit_predict(X)

    # the silhouette_score gives the average value for all the samples #
    # this gives a perspective into the density and separation of the formed #
    # clusters #
    silhouette_avg = silhouette_score(X, cluster_labels)
    print(
        "For n_clusters =",
        n_clusters,
        "The average silhouette_score is :",
        silhouette_avg,
    )

    # compute the silhouette scores for each sample #
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # aggregate the silhouette scores for samples belonging to #
        # cluster i, and sort them #
        ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(
            np.arange(y_lower, y_upper),
            0,
            ith_cluster_silhouette_values,
            facecolor=color,
            edgecolor=color,
            alpha=0.7,
        )

        # label the silhouette plots with their cluster numbers at the middle #
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # compute the new y_lower for next plot #
        y_lower = y_upper + 10  # 10 for the 0 samples #

    ax1.set_title("The silhouette plot for the various clusters")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # the vertical line for average silhouette score of all the values #
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # clear the yaxis labels / ticks #
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd plot showing the actual clusters formed #
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(
        X[:, 0], X[:, 1], marker=".", s=30, lw=0, alpha=0.7, c=colors, edgecolor="k"
    )

    # labeling the clusters #
    centers = clusterer.cluster_centers_
    # draw white circles at cluster centers #
    ax2.scatter(
        centers[:, 0],
        centers[:, 1],
        marker="o",
        c="white",
        alpha=1,
        s=200,
        edgecolor="k",
    )

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker="$%d$" % i, alpha=1, s=50, edgecolor="k")

    ax2.set_title("The visualization of the clustered data")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(
        "Silhouette Analysis for K-means Clustering on Exoplanet Data With n_clusters = %d"
        % n_clusters,
        fontsize=14,
        fontweight="bold",
    )

plt.show()
```

```
For n_clusters = 2 The average silhouette_score is : 0.6384634701375508
For n_clusters = 3 The average silhouette_score is : 0.5429813171902624
For n_clusters = 4 The average silhouette_score is : 0.5635026172929426
For n_clusters = 5 The average silhouette_score is : 0.5516999350815642
For n_clusters = 6 The average silhouette_score is : 0.5519380483842715
```

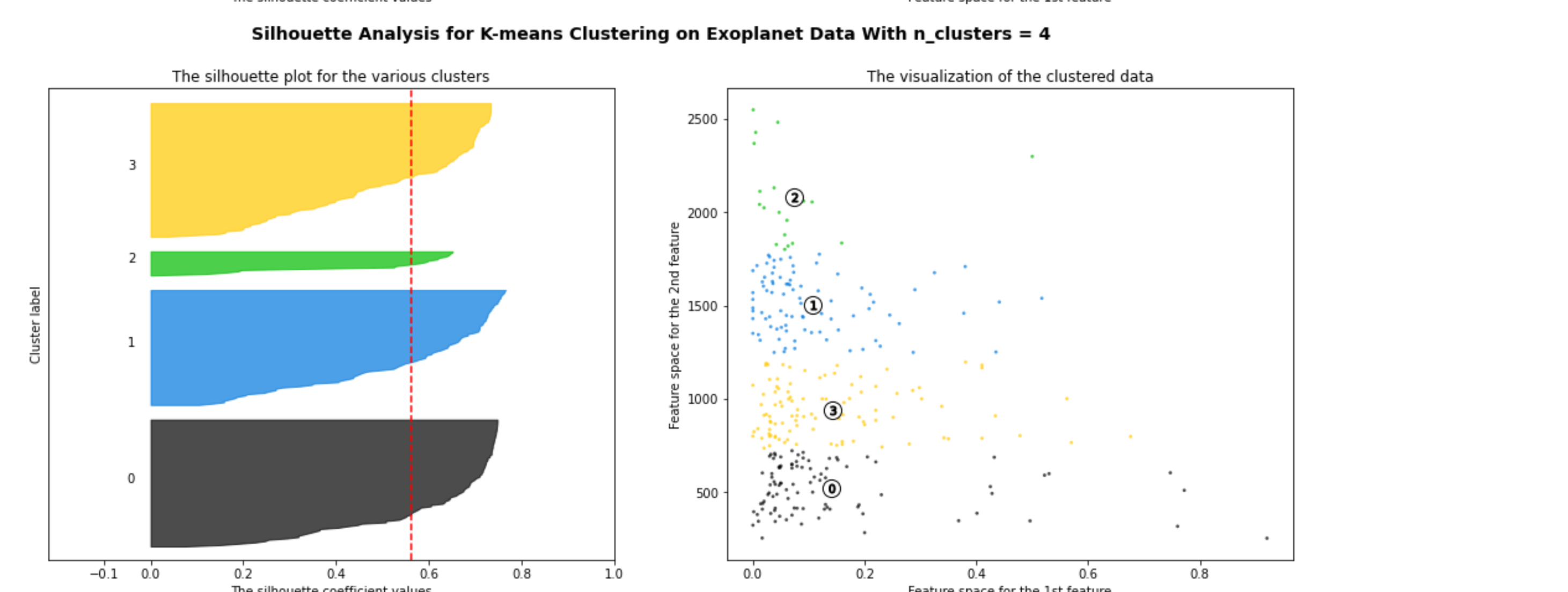
Silhouette Analysis for K-means Clustering on Exoplanet Data With n_clusters = 2



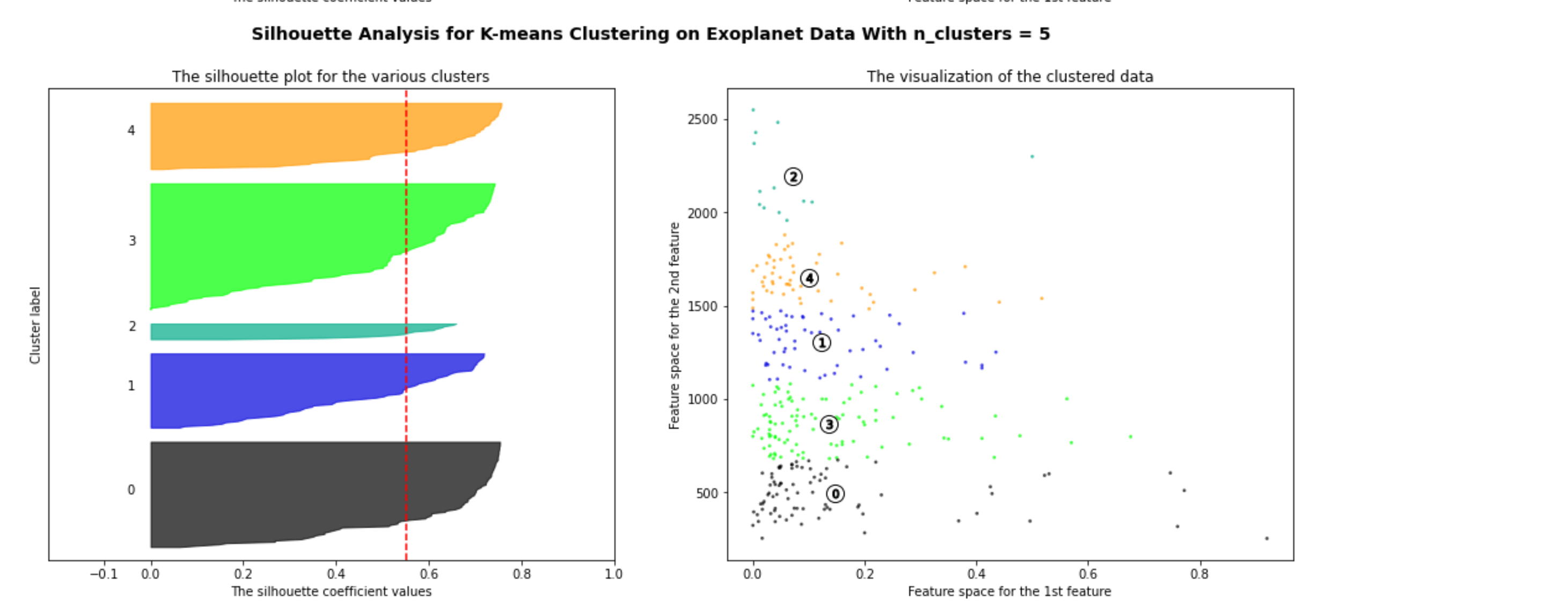
Silhouette Analysis for K-means Clustering on Exoplanet Data With n_clusters = 3



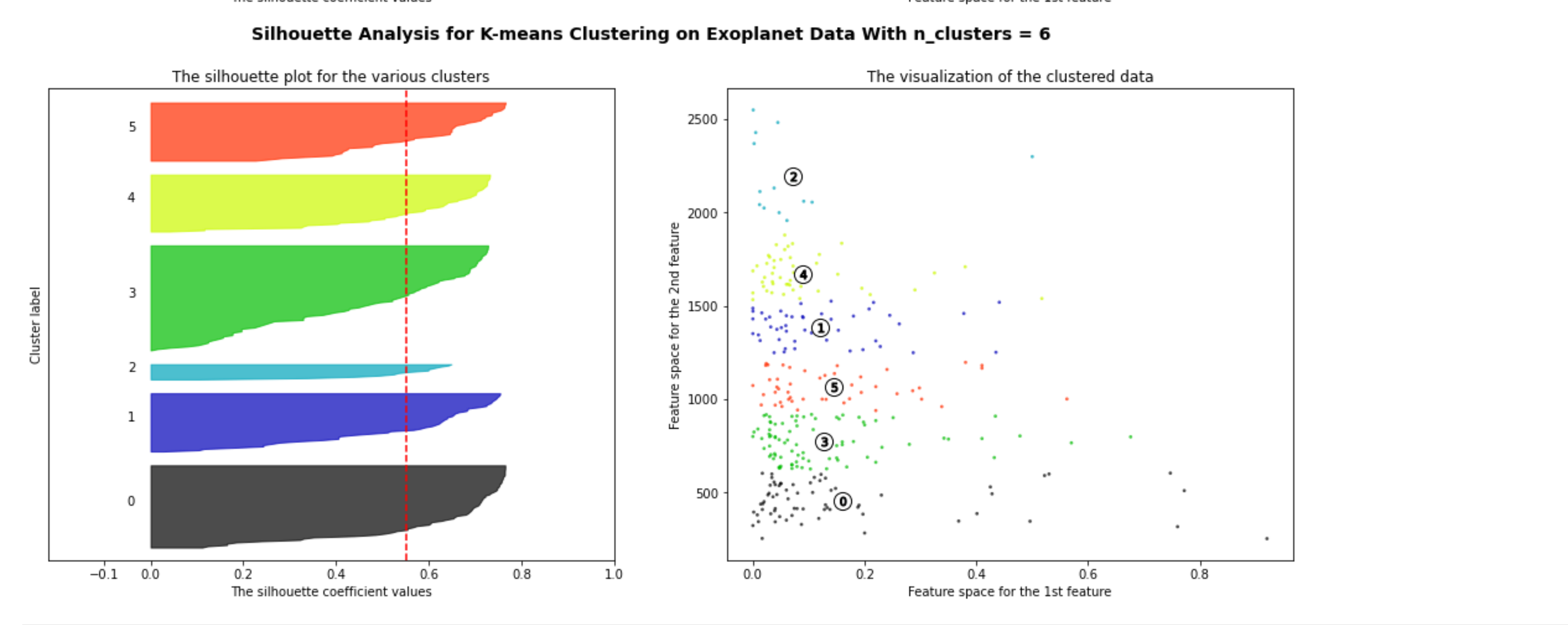
Silhouette Analysis for K-means Clustering on Exoplanet Data With n_clusters = 4



Silhouette Analysis for K-means Clustering on Exoplanet Data With n_clusters = 5



Silhouette Analysis for K-means Clustering on Exoplanet Data With n_clusters = 6



In [17]:
```
# add pca results to the original dataframe #
df['PC1'] = pca_result[:, 0]
df['PC2'] = pca_result[:, 1]
```

In [18]:
```
df.tail()
```

Out[18]:

	pl_name	pl_rade	pl_masse	pl_orbeccen	pl_eqt	PC1	PC2
34655	wasp-80 b	11.198	170.99254	0.0020	825.0	-0.233537	-0.672940
34703	wasp-89 b	11.657	1875.19700	0.1930	1120.0	1.250454	0.773264
34785	wolf 503 b	2.043	6.26000	0.4100	790.0	-0.969503	1.782680
34854	xo-7 b	15.390	225.34147	0.0380	1743.0	1.369523	-1.333657
328	earth	1.000	1.00000	0.0167	255.0	-2.119478	0.059587

K-means Clustering

In [19]:
```
# number of clusters #
num_clusters = 3

# initialize kmeans model #
kmeans = KMeans(n_clusters=num_clusters, random_state=42)

# fit the model to main dataset #
kmeans.fit(df[features])
```

Out[19]:

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

In [20]:
```
# predict clusters #
df['cluster'] = kmeans.predict(df[features])
```

Plot Clusters

Earth marked with a star within its cluster

In [21]:
```
# create a scatter plot with plotly #
fig = px.scatter(df, x='PC1', y='PC2', color='cluster',
    color_continuous_scale = 'portland', title='K-means Clustering With PCA')

# select the earth trace to mark with '*' #
trace_to_mark = 302

# add a new trace with '*' marker #
marked_trace = px.scatter(df.iloc[trace_to_mark:trace_to_mark + 1], x='PC1', y='PC2',
                          color='cluster', color_continuous_scale = 'portland',
                          title='K-means Clustering With PCA')
marked_trace.update_traces(marker=dict(symbol='star', size=15))

# add the new marked trace to the existing figure #
fig.add_trace(marked_trace.data[0])

# calculate cluster centroids #
centroids = df.groupby('cluster')['PC1', 'PC2'].mean().reset_index()

# add centroids as markers #
centroid_trace = px.scatter(centroids, x='PC1', y='PC2', color='cluster',
                            color_continuous_scale='portland')
centroid_trace.update_traces(marker=dict(symbol='diamond', size=10, color='black'))

fig.add_trace(centroid_trace.data[0])

# update x and y axis ranges #
# x_range = [-3, 8]  # replace xmin and xmax with desired range #
# y_range = [-5, 6]  # replace ymin and ymax with desired range #
# fig.update_layout(xaxis_range=x_range, yaxis_range=y_range) #

fig.show()
```