Thomas Kosiolek

1.

```
import pandas as pd

df = pd.read_csv('customer.csv')

df
```

	cust_key_value	cust_freq_address	cust_freq_pf	cust_freq_tel	cust_freq_cnic	cust_freq_mobile	cust_freq_internet	cust_freq_cricket	cust_freq_gaming	...	chg_freq_fixl	chg_freq_ftv
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00
1	0.01	0.01	0.00	0.01	0.01	0.00	0.01	0.07	0.00	...	0.00	0.01
2	0.06	0.09	0.71	0.0	1.23	0.14	0.00	0.12	0.64	...	0.00	0.0
3	0.00	0.00	0.00	0.0	0.00	0.00	0.01	0.00	0.01	...	0.00	0.00
4	0.00	0.00	0.00	0.0	0.00	0.00	0.01	0.00	0.01	...	0.00	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
2994	0.01	0.00	0.02	0.0	0.00	0.01	0.00	0.00	0.00	...	0.00	0.00
2995	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00
2996	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	...	0.00	0.0
2997	0.06	0.00	0.00	0.0	0.02	0.00	0.00	0.00	0.00	...	0.00	0.00
2998	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	...	0.00	0.0

4823 rows × 50 columns

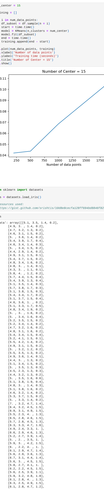
```
for i in range(len(df):
    for column in df.columns:
        if df[column].isnull().any():
            print(f"{column}: {df[column].isnull().sum()}")
```

```
# Resources used:
# https://www.geeksforgeeks.org/dataclysm-pattern-recognition/dataclysm-pattern-recognition-using-numpy-and-pandas/#:~:text=Pandas%20is%20a%20powerful,%2C%20and%20much%20more.&text=Pandas%20%2D%20Dataclysm%20Pattern%20Recognition,-Dataclysm%20Pattern%20Recognition%20is%20a%20powerful%20and%20versatile%20tool%20for%20numerical%20and%20time%20series%20analysis.
```

```
# Running p_check
df.shape
for column in df.columns:
    print(f"{column}: {df[column].dtype}")
    if df[column].isnull().any():
        print("True")

# Resources used:
# https://stackoverflow.com/questions/40039689/how-to-use-the-if-all-statement-in-python

All of the columns are numeric.

num_data_points = [100, 500, 1000, 2000]
num_centers = [2, 5, 10, 15, 20]
```

```
from sklearn.cluster import KMeans
import time

training = []

for i in num_centers:
    print("Clustering for", i)
    model = KMeans(n_clusters = i)
    model.fit(df)
    print("Done")
    training.append(end - start)

# Resources used:
# https://www.geeksforgeeks.org/how-to-find-optimal-clusters-in-keras-keras-and-tensorflow-and-pytorch/#:~:text=Finding%20the%20optimal%20number%20of,%2C%20Keras%20%2C%20TensorFlow%20%2C%20and%20PyTorch.
# https://www.geeksforgeeks.org/ml-hierarchical-clustering-agglomerative-and-divisive-clustering/
# https://www.programiz.com/python-programming/examples/kilometer-time
```

```
import matplotlib.pyplot as plt

plt.plot(num_centers, training)
plt.xlabel('Number of centers')
plt.ylabel('Training time (seconds)')
plt.show()
```




```
from sklearn.metrics import silhouette_score

cluster_score = []

for i in num_centers:
    print("Clustering for", i)
    model.fit(df)
    labels = model.labels_
    score = silhouette_score(df, labels)
    cluster_score.append(score)

plt.plot(num_centers, cluster_score)
plt.xlabel('Number of centers')
plt.ylabel('Clustering Score')
plt.show()

# Resources used:
# https://towardsdatascience.com/optimizing-kmeans-silhouette-clustering-part-3-clustering-a8b6b9b8a3dc#:~:text=Parameter%20%2D%20Number%20of,Score%20%2D%20Silhouette%20Score
# https://stackoverflow.com/questions/51382839/silhouette-score-explained-with-python-example/
```

```
num_center = 15

training = []

for i in num_data_points:
    print("Clustering for", i)
    start = time.time()
    model = KMeans(n_clusters = num_center)
    model.fit(df)
    end = time.time()
    training.append(end - start)

plt.plot(num_data_points, training)
plt.xlabel('Number of data points')
plt.ylabel('Training time (seconds)')
plt.title('Number of Center = 15')
plt.show()
```

2.

```
from sklearn import datasets

iris = datasets.load_iris()

# Resources used:
# https://plot.ly/ml-in-1-line/#/clustering/k-means-clustering
```

```
iris
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3.0, 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5.0, 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5.0, 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3.0, 1.4, 0.1],
        [4.3, 3.0, 1.1, 0.1],
        [5.8, 4.0, 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
        [5.1, 3.8, 1.5, 0.3],
        [5.4, 3.4, 1.7, 0.2],
        [5.1, 3.7, 1.5, 0.4],
        [4.6, 3.6, 1.0, 0.2],
        [5.1, 3.3, 1.7, 0.5],
        [4.8, 3.4, 1.9, 0.2],
        [5.0, 3.0, 1.6, 0.2],
        [5.0, 3.4, 1.6, 0.4],
        [5.2, 3.5, 1.5, 0.2],
        [5.2, 3.4, 1.4, 0.2],
        [4.7, 3.2, 1.6, 0.2],
        [4.8, 3.1, 1.6, 0.2],
        [5.4, 3.4, 1.5, 0.4],
        [5.2, 4.1, 1.5, 0.1],
        [5.5, 4.2, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.0, 3.2, 1.2, 0.2],
        [5.5, 3.5, 1.3, 0.2],
        [4.9, 3.6, 1.4, 0.1],
        [4.4, 3.0, 1.3, 0.2],
        [5.1, 3.4, 1.5, 0.2],
        [5.0, 3.5, 1.3, 0.3],
        [4.5, 2.3, 1.3, 0.3],
        [4.4, 3.2, 1.3, 0.2],
        [5.0, 3.5, 1.6, 0.6],
        [5.1, 3.8, 1.9, 0.4],
        [4.8, 3.0, 1.4, 0.3],
        [5.1, 3.8, 1.6, 0.2],
        [4.6, 3.2, 1.4, 0.2],
        [5.3, 3.7, 1.5, 0.2],
        [5.0, 3.3, 1.4, 0.2],
        [7.0, 3.2, 4.7, 1.4],
        [6.4, 3.2, 4.5, 1.5],
        [6.9, 3.1, 4.9, 1.5],
        [5.5, 2.3, 4.0, 1.3],
        [6.5, 2.8, 4.6, 1.5],
        [5.7, 2.8, 4.5, 1.3],
        [6.3, 3.3, 4.7, 1.6],
        [4.9, 2.4, 3.3, 1.0],
        [6.6, 2.9, 4.6, 1.3],
        [5.2, 2.7, 3.9, 1.4],
        [5.0, 2.0, 3.5, 1.0],
        [5.9, 3.0, 4.2, 1.5],
        [6.0, 2.2, 4.0, 1.0],
        [6.1, 2.9, 4.7, 1.4],
        [5.6, 2.9, 3.6, 1.3],
        [6.7, 3.1, 4.4, 1.4],
        [5.6, 3.0, 4.5, 1.5],
        [5.8, 2.7, 4.1, 1.0],
        [6.2, 2.2, 4.5, 1.5],
        [5.6, 2.5, 3.9, 1.1],
        [5.9, 3.2, 4.8, 1.8],
        [6.1, 2.8, 4.0, 1.3],
        [6.3, 2.5, 4.9, 1.5],
        [6.1, 2.8, 4.7, 1.2],
        [6.4, 2.9, 4.3, 1.3],
        [6.6, 3.0, 4.4, 1.4],
        [6.8, 2.8, 4.8, 1.4],
        [6.7, 3.0, 5.0, 1.7],
        [6.0, 2.9, 4.5, 1.5],
        [5.7, 2.6, 3.5, 1.0],
        [5.5, 2.4, 3.8, 1.1],
        [5.5, 2.4, 3.7, 1.0],
        [5.8, 2.7, 3.9, 1.2],
        [6.0, 2.7, 5.1, 1.6],
        [5.4, 3.0, 4.5, 1.5],
        [6.0, 3.4, 4.5, 1.6],
        [6.7, 3.1, 4.7, 1.5],
        [6.3, 2.3, 4.4, 1.3],
        [5.6, 3.0, 4.1, 1.3],
        [5.5, 2.5, 4.0, 1.3],
        [5.5, 2.6, 4.4, 1.2],
        [6.1, 3.0, 4.6, 1.4],
        [5.8, 2.6, 4.0, 1.2],
        [5.0, 2.3, 3.3, 1.0],
        [5.6, 2.7, 4.2, 1.3],
        [5.7, 3.0, 4.2, 1.2],
        [5.7, 2.9, 4.2, 1.3],
        [6.2, 2.9, 4.3, 1.3],
        [5.1, 2.5, 3.0, 1.1],
        [5.7, 2.8, 4.1, 1.3],
        [6.3, 3.3, 6.0, 2.5],
        [5.8, 2.7, 5.1, 1.9],
        [7.1, 3.0, 5.9, 2.1],
        [6.3, 2.9, 5.6, 1.8],
        [6.5, 3.0, 5.8, 2.2],
        [7.6, 3.0, 6.6, 2.1],
        [4.9, 2.5, 4.5, 1.7],
        [7.3, 2.9, 6.3, 1.8],
        [6.7, 2.5, 5.8, 1.8],
        [7.2, 3.6, 6.1, 2.5],
        [6.5, 3.2, 5.1, 2.0],
        [6.4, 2.7, 5.3, 1.9],
        [6.8, 3.0, 5.5, 2.1],
        [5.7, 2.5, 5.0, 2.0],
        [5.8, 2.8, 5.1, 2.4],
        [6.4, 3.2, 5.3, 2.3],
        [6.5, 3.0, 5.5, 1.8],
        [7.7, 3.8, 6.7, 2.2],
        [7.7, 2.6, 6.9, 2.3],
        [6.0, 2.2, 5.0, 1.5],
        [6.9, 3.2, 5.7, 2.3],
        [5.6, 2.8, 4.9, 2.0],
        [7.7, 2.8, 6.7, 2.0],
        [6.3, 2.7, 4.9, 1.8],
        [6.7, 3.3, 5.7, 2.1],
        [7.2, 3.2, 6.0, 1.8],
        [6.2, 2.8, 4.8, 1.8],
        [6.1, 3.0, 4.9, 1.8],
        [6.4, 2.8, 5.6, 2.1],
        [7.2, 3.0, 5.8, 1.6],
        [7.4, 2.8, 6.1, 1.9],
        [7.9, 3.8, 6.4, 2.0],
        [6.4, 2.8, 5.6, 2.2],
        [6.3, 2.8, 5.1, 1.5],
        [6.1, 2.6, 5.6, 1.4],
        [7.7, 3.0, 6.1, 2.3],
        [6.3, 3.4, 5.6, 2.4],
        [6.4, 3.1, 5.5, 1.8],
        [6.0, 3.0, 4.8, 1.8],
        [6.9, 3.1, 5.4, 2.1],
        [6.7, 3.1, 5.6, 2.4],
        [6.9, 3.1, 5.1, 2.3],
        [5.8, 2.7, 5.1, 1.9],
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3.0, 5.2, 2.3],
        [6.3, 2.5, 5.0, 1.9],
        [6.5, 3.0, 5.2, 2.0],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3.0, 5.1, 1.8]],
       dtype=float64),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
```