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

(1)

In [2]:
```
csv_in = '2024-cs3-mid-1.csv'
df = pd.read_csv(csv_in, skiprows=1, sep=',', header=0)
```

(2)(3)(4)

In [3]:
```
print(df.shape)
print(df.info())
display(df.head())
```

```
(40, 7)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ID      40 non-null     int64  
 1   c1      40 non-null     float64
 2   c2      40 non-null     float64
 3   c3      40 non-null     float64
 4   c4      40 non-null     float64
 5   c5      40 non-null     float64
 6   c6      40 non-null     object 
dtypes: float64(5), int64(1), object(1)
memory usage: 2.3+ KB
None
```

	ID	c1	c2	c3	c4	c5	c6
0	0	1.849665	3.520478	2.644960	3.772676	1.100279	b
1	1	1.424760	2.603061	1.458631	4.419606	4.569825	b
2	2	1.986131	4.493088	4.829959	3.479897	2.838382	d
3	3	0.337362	4.150143	-0.371423	1.699369	3.908376	e
4	4	2.330581	1.510337	4.955510	2.810172	2.921209	b

(5) 40

(6) 7

(7) (8)

In [4]:
```
ser_c1 = df['c1']
print(ser_c1.describe())
```

```
count    40.000000
mean      2.585942
std       1.463949
min      -0.264668
25%       1.497987
50%       2.737563
75%       3.389756
max       5.923593
Name: c1, dtype: float64
```

(9)

2.74

(10)

In [5]:
```
display(df.sort_values(by='c1',ascending=False).head())
```

	ID	c1	c2	c3	c4	c5	c6
35	35	5.923593	2.868861	1.826152	0.913487	0.619175	c
21	21	5.115733	2.808015	4.687095	4.222077	3.284643	c
25	25	4.923591	5.624131	1.773394	2.437812	4.997738	a
14	14	4.867156	0.425360	1.984590	1.620037	1.608496	d
15	15	4.765931	5.093854	3.245133	3.416734	4.727864	f

(11)

5.12

(12)

In [6]:
```
print( df['c6'].value_counts() )
```

```
c6
d    10
b     7
e     7
f     7
c     6
a     3
Name: count, dtype: int64
```

(13)

10

(14) (15)

In [7]:
```
df2=df.drop(columns=['ID','c4','c5'])
```

In [8]:
```
display(df2.groupby('c6').mean())
```

c6	c1	c2	c3
a	3.235814	3.383295	3.052760
b	2.300865	2.641799	3.163440
c	3.421703	3.432855	2.442878
d	1.989892	3.338549	2.699112
e	2.196615	3.492857	2.332375
f	3.116962	3.823327	3.480458

(16)

3.49

(17)(18)

In [9]:
```
df['n_tot']=df['c1']+df['c2']+df['c3']
df3=df.sort_values(by='n_tot', ascending=False)
display(df3.head())
```

	ID	c1	c2	c3	c4	c5	c6	n_tot
5	5	3.143336	5.027989	5.567861	3.488507	1.877655	e	13.739186
15	15	4.765931	5.093854	3.245133	3.416734	4.727864	f	13.104917
21	21	5.115733	2.808015	4.687095	4.222077	3.284643	c	12.610842
25	25	4.923591	5.624131	1.773394	2.437812	4.997738	a	12.321116
29	29	2.968940	3.414144	5.650455	0.891602	1.395094	f	12.033538

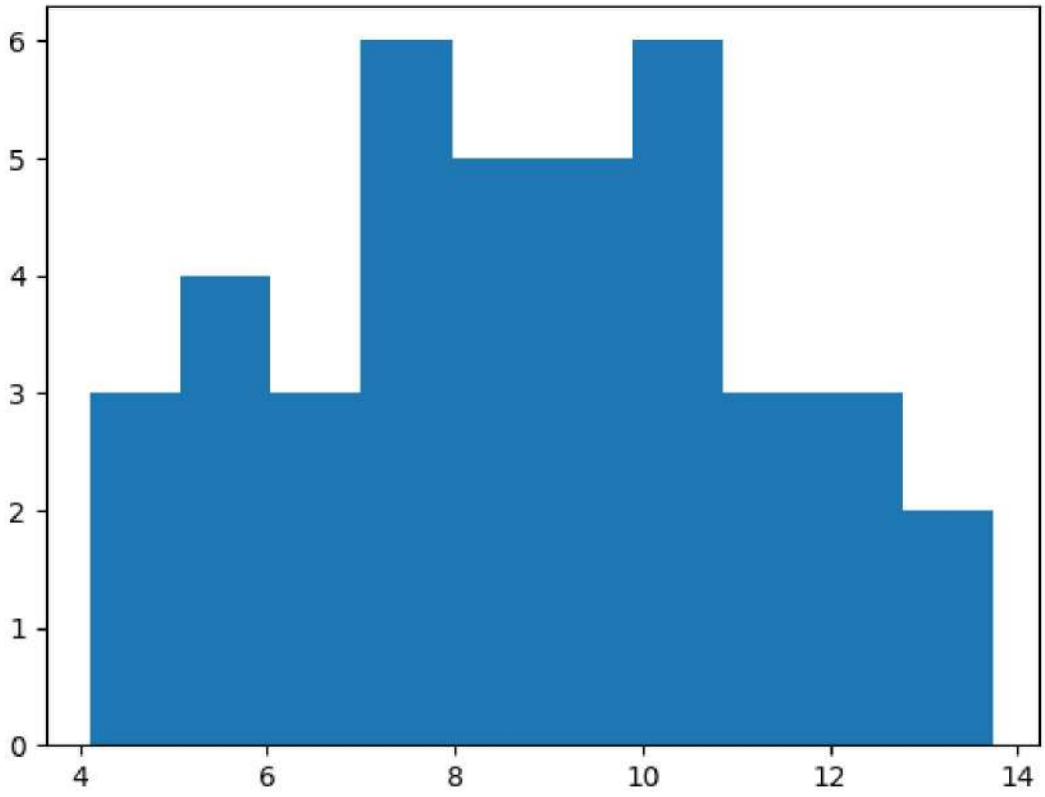
(19)

13.74

(20)

In [10]:
```
plt.hist(df3['n_tot'], bins=10)
plt.plot()
```

Out[10]:
```
[]
```


6
5
4
3
2
1
0
4
6
8
10
12
14

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
csv_in = '2024-cs3-mid-2-1.csv'
csv_in2 = '2024-cs3-mid-2-2.csv'
```

In [4]:
```
df1 = pd.read_csv(csv_in, skiprows=0, sep=',', header=0)
df2 = pd.read_csv(csv_in2, skiprows=0, sep=',', header=0)
```

In [5]:
```
print(df1.shape)
print(df1.info())
display(df1.head())
```

```
(50, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      50 non-null     int64
 1   c1      50 non-null     float64
 2   c2      47 non-null     float64
 3   c3      49 non-null     float64
 4   q1      50 non-null     object
dtypes: float64(3), int64(1), object(1)
memory usage: 2.1+ KB
None
```

	ID	c1	c2	c3	q1
0	0	-0.787345	3.754889	0.876407	c
1	1	-0.119684	3.620732	1.944058	b
2	2	-0.656145	4.336709	1.352879	c
3	3	-2.012349	5.430930	0.506446	c
4	4	1.283513	5.497184	3.872931	d

In [6]:
```
print(df2.shape)
print(df2.info())
display(df2.head())
```

```
(50, 2)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   IDX     50 non-null     int64
 1   q2      50 non-null     object
dtypes: int64(1), object(1)
memory usage: 928.0+ bytes
None
```

	IDX	q2
0	0	t
1	1	t
2	2	s
3	3	r
4	4	r

(1)

In [7]:
```
display(df1[df1.duplicated(keep=False)])
```

	ID	c1	c2	c3	q1
7	7	1.050161	2.207037	2.180943	c
36	7	1.050161	2.207037	2.180943	c
43	7	1.050161	2.207037	2.180943	c

(2)

In [8]:
```
df1m = df1.drop_duplicates().reset_index(drop=True)
```

In [9]:
```
print(df1m.shape)
```

```
(48, 5)
```

(3)

48

(4)

In [10]:
```
print( df1m.isna().sum(axis=0) )
```

```
ID    0
c1    0
c2    3
c3    1
q1    0
dtype: int64
```

(5)

c2, c3

(6)

In [11]:
```
display( df1m[df1m.isnull().any(axis=1)] )
```

	ID	c1	c2	c3	q1
10	10	0.546855	NaN	3.409160	c
31	31	1.388495	NaN	NaN	c
43	45	0.436719	NaN	1.853421	b

(7)

In [12]:
```
df1m2 = df1m.dropna().reset_index(drop=True)
```

(8)

In [13]:
```
print( df2['q2'].value_counts() )
```

```
q2
t    18
s    16
r    16
Name: count, dtype: int64
```

(9)

18

(10)

In [14]:
```
df3=pd.merge(df1m2, df2, left_on='ID', right_on='IDX', how='inner')
display(df3.head())
```

	ID	c1	c2	c3	q1	IDX	q2
0	0	-0.787345	3.754889	0.876407	c	0	t
1	1	-0.119684	3.620732	1.944058	b	1	t
2	2	-0.656145	4.336709	1.352879	c	2	s
3	3	-2.012349	5.430930	0.506446	c	3	r
4	4	1.283513	5.497184	3.872931	d	4	r

(11)

In [15]:
```
df3_ret = df3[ (df3['q2']=='s') & (df3['c1']>2.0) ]
display(df3_ret)
```

	ID	c1	c2	c3	q1	IDX	q2
13	14	2.473574	4.858444	1.621829	c	14	s
28	29	3.833894	3.370421	0.947914	a	29	s
36	39	2.110630	5.401511	-0.217790	d	39	s
38	41	2.013292	4.047493	5.115328	d	41	s
43	48	2.333132	3.117964	2.955144	c	48	s

(12)

In [16]:
```
df3_ret.to_csv('2024-cs3-mid-out.csv', index=False)
```

In [17]:
```
df3_ret.shape
```

Out[17]:
```
(5, 7)
```

(13)(14)

5, 7

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

In [2]:
```
csv_in = 'google-stock-price.csv'
df = pd.read_csv(csv_in, sep=',', skiprows=0, header=0)
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1100 entries, 0 to 1099
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   date    1100 non-null   object 
 1   price   1100 non-null   float64
dtypes: float64(1), object(1)
memory usage: 17.3+ KB
```

Out[2]:

	date	price
0	1/2/2020	68.37
1	1/3/2020	68.03
2	1/6/2020	69.71
3	1/7/2020	69.67
4	1/8/2020	70.22

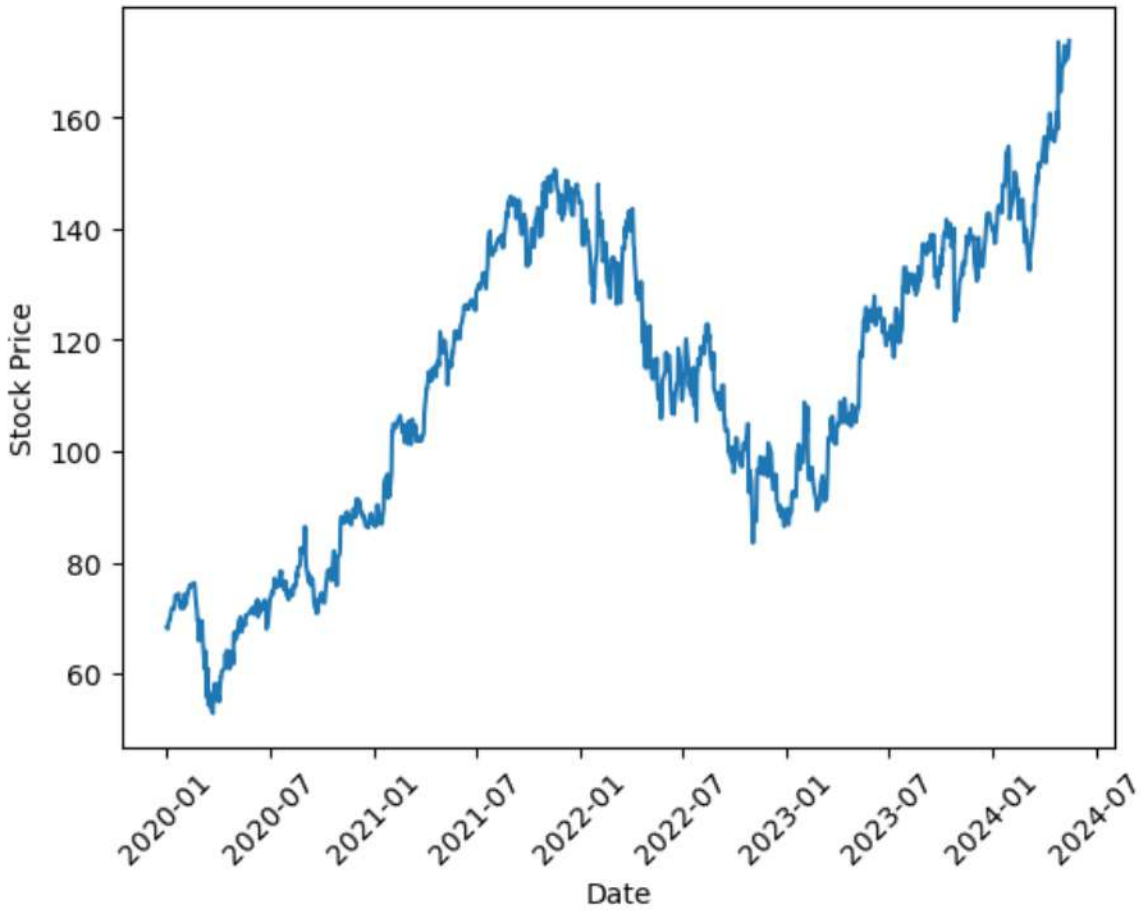
In [3]:
```
df['date'] = pd.to_datetime(df['date']) #1
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1100 entries, 0 to 1099
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    1100 non-null   datetime64[ns]
 1   price   1100 non-null   float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 17.3 KB
None
```

In [4]:
```
df = df.set_index('date') #2
display(df.head())
```

	price
date	
2020-01-02	68.37
2020-01-03	68.03
2020-01-06	69.71
2020-01-07	69.67
2020-01-08	70.22

In [5]:
```
plt.plot(df.index, df["price"])
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.xticks(rotation=45)
plt.show()
```

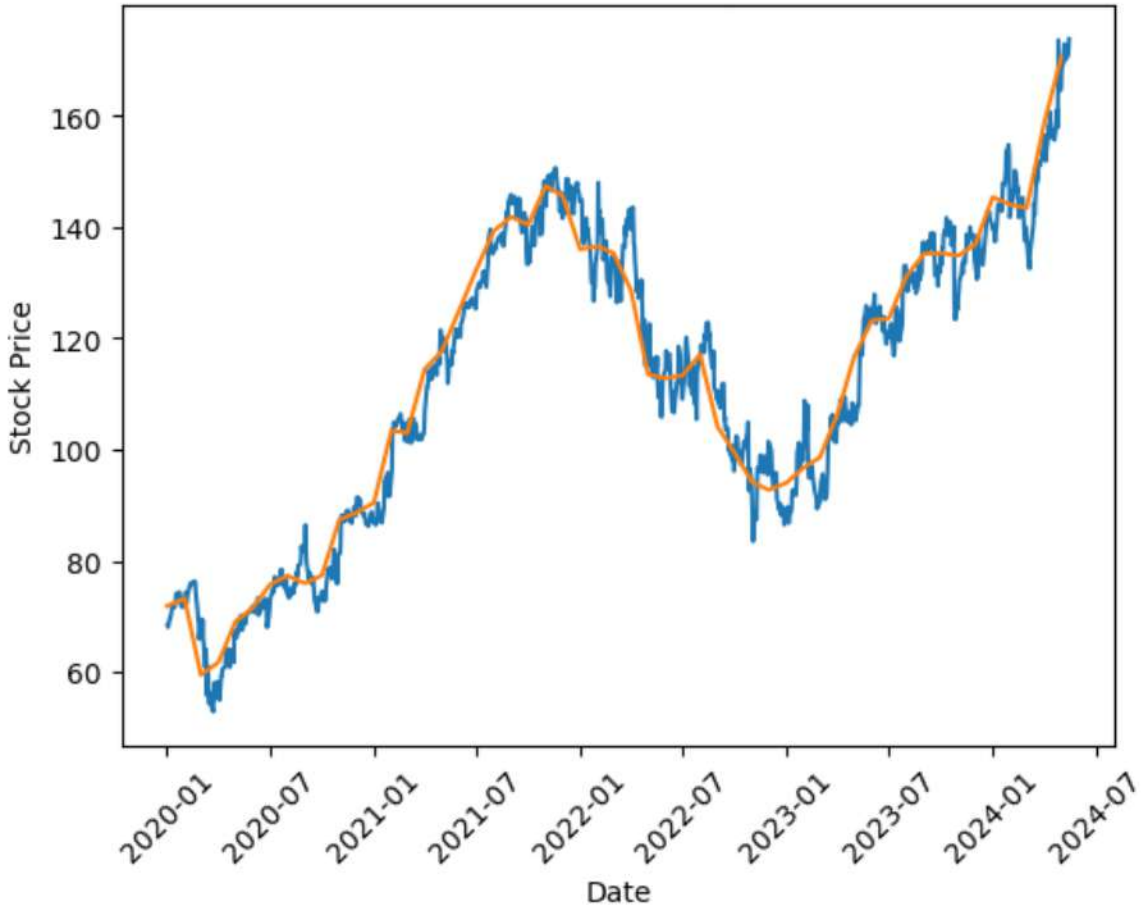


In [6]:
```
df_month_start = df.resample('MS').mean() #3
display(df_month_start.head())
display(df_month_start.tail())
```

	price
date	
2020-01-01	71.832381
2020-02-01	73.204737
2020-03-01	59.420455
2020-04-01	61.706190
2020-05-01	69.056500

	price
date	
2024-01-01	145.425714
2024-02-01	144.068000
2024-03-01	143.481500
2024-04-01	158.730909
2024-05-01	170.506364

In [7]:
```
plt.plot(df.index, df["price"])
plt.plot(df_month_start.index, df_month_start["price"])
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.xticks(rotation=45)
plt.show()
```



In [8]:
```
df['remark'] = ['high' if x>100 else 'low' for x in df['price']] #4
df['name_of_day'] = df.index.day_name() #5
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1100 entries, 2020-01-02 to 2024-05-15
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   price        1100 non-null   float64
 1   remark       1100 non-null   object 
 2   name_of_day  1100 non-null   object 
dtypes: float64(1), object(2)
memory usage: 34.4+ KB
```

Out[8]:

date	price	remark	name_of_day
2020-01-02	68.37	low	Thursday
2020-01-03	68.03	low	Friday
2020-01-06	69.71	low	Monday
2020-01-07	69.67	low	Tuesday
2020-01-08	70.22	low	Wednesday

In [9]:
```
df_23 = df[df.index.year == 2023].copy() # 6, 7
df_23.info() # 8 (Ans: 250行)
```

```
display(df_23.head())
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 250 entries, 2023-01-03 to 2023-12-29
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   price        250 non-null    float64
 1   remark       250 non-null    object 
 2   name_of_day  250 non-null    object 
dtypes: float64(1), object(2)
memory usage: 7.8+ KB
```

date	price	remark	name_of_day
2023-01-03	89.70	low	Tuesday
2023-01-04	88.71	low	Wednesday
2023-01-05	86.77	low	Thursday
2023-01-06	88.16	low	Friday
2023-01-09	88.80	low	Monday

In [10]:
```
df_23_ctab = pd.crosstab(df_23['name_of_day'], df_23['remark'], margins=True) #9
display(df_23_ctab)
```

name_of_day	remark	high	low	All
Friday		43	8	51
Monday		39	6	45
Thursday		42	9	51
Tuesday		41	10	51
Wednesday		42	10	52
All		207	43	250

In [1]:
```
# To avoid the warning about the memory leak of KMeans (Win MKL version)
import os
os.environ['OMP_NUM_THREADS'] = '1'

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import scale
```

In [2]:
```
# To show all rows/columns
pd.options.display.max_columns = 999
pd.options.display.max_rows = 999
```

In [3]:
```
csv_in = 'driver-data.csv'
df = pd.read_csv(csv_in, sep=',', skiprows=0, header=0)
print(df.shape)
print(df.info())
display(df.head())
display(df.tail())
```

```
(4000, 3)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   driver_id       4000 non-null   int64
 1   avg_dist_day    4000 non-null   float64
 2   avg_over_speed  4000 non-null   int64
dtypes: float64(1), int64(2)
memory usage: 93.9 KB
None
```

	driver_id	avg_dist_day	avg_over_speed
0	3423311935	71.24	28
1	3423313212	52.53	25
2	3423313724	64.54	27
3	3423311373	55.69	22
4	3423310999	54.58	25

	driver_id	avg_dist_day	avg_over_speed
3995	3423310685	160.04	10
3996	3423312600	176.17	5
3997	3423312921	170.91	12
3998	3423313630	176.14	5
3999	3423311533	168.03	9

In [4]:
```
dfX = df[['avg_dist_day', 'avg_over_speed']] #1
print(dfX.shape)
display(dfX.head())
```

```
(4000, 2)
```

	avg_dist_day	avg_over_speed
0	71.24	28
1	52.53	25
2	64.54	27
3	55.69	22
4	54.58	25

In [5]:
```
X_scaled = scale(dfX) #2
```

In [6]:
```
print(type(X_scaled))
print(X_scaled.shape)
```

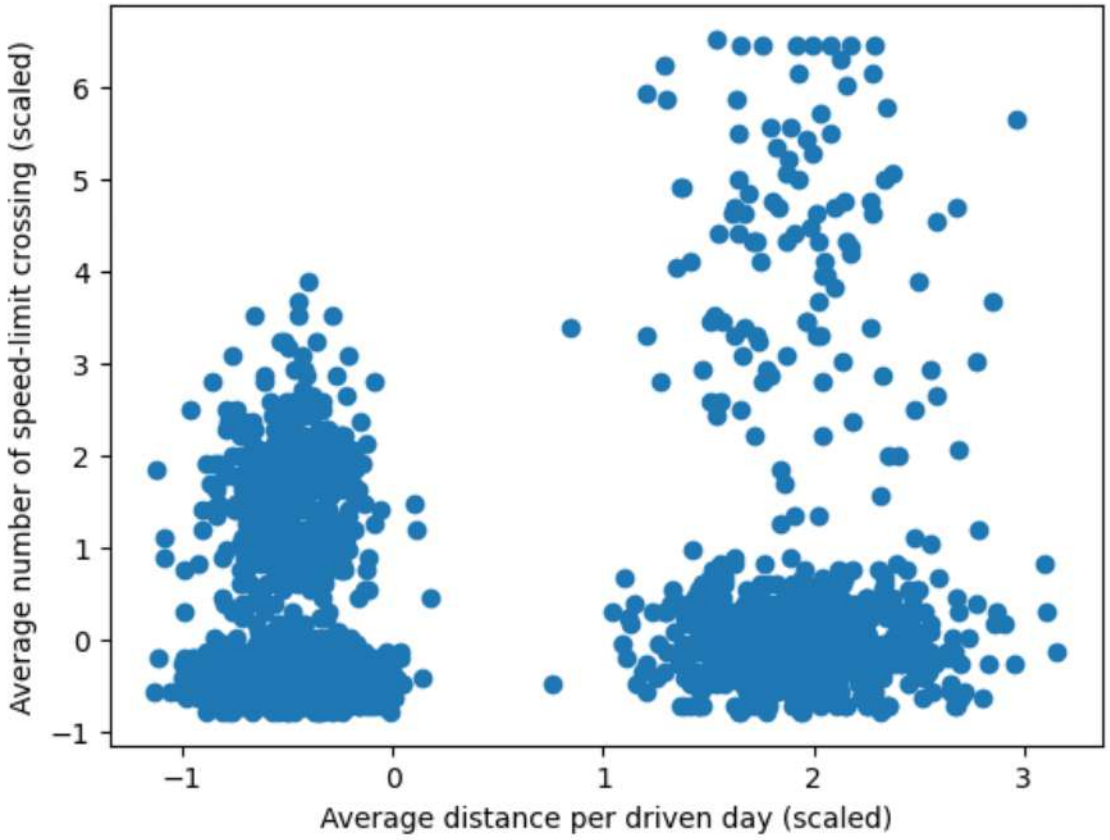
```
<class 'numpy.ndarray'>
(4000, 2)
```

In [7]:
```
dfX_scaled = pd.DataFrame(X_scaled, columns=dfX.columns) #3
print(type(dfX_scaled))
display(dfX_scaled.head())
```

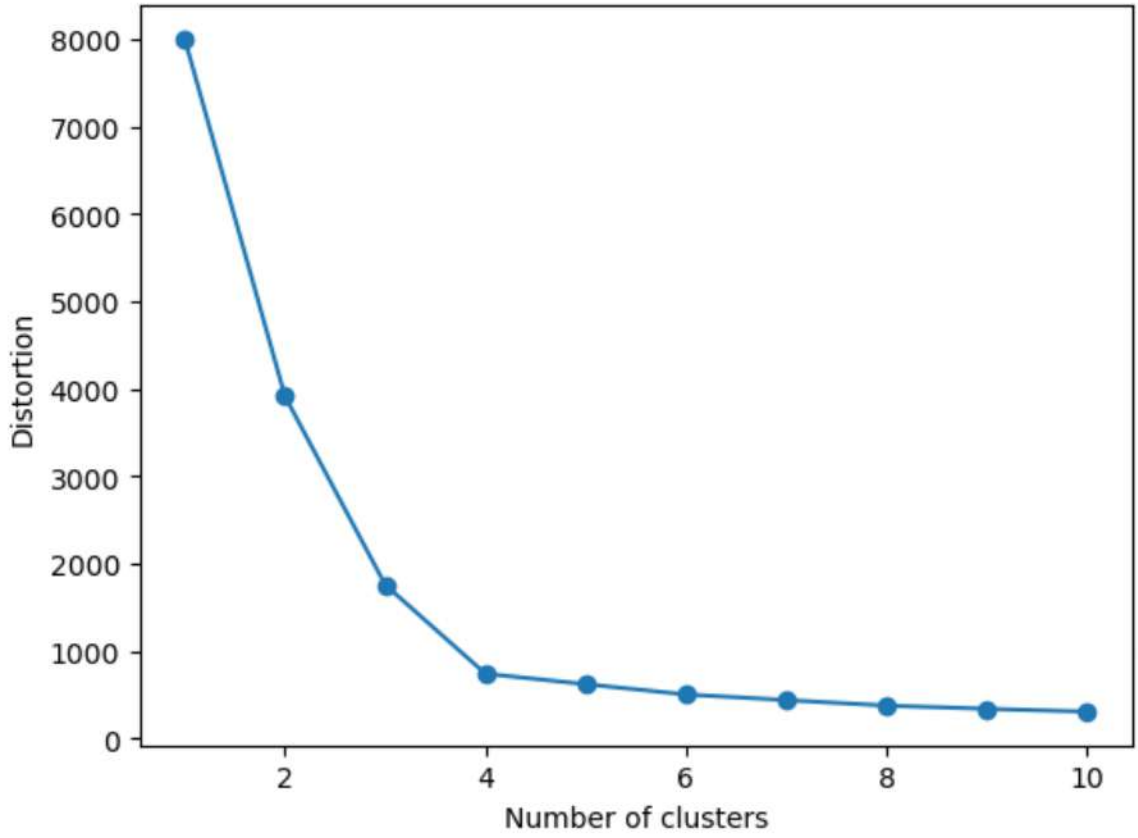
```
<class 'pandas.core.frame.DataFrame'>
```

	avg_dist_day	avg_over_speed
0	-0.089810	1.260613
1	-0.439773	1.041744
2	-0.215131	1.187656
3	-0.380666	0.822875
4	-0.401428	1.041744

In [8]:
```
plt.scatter(dfX_scaled['avg_dist_day'], dfX_scaled['avg_over_speed'], marker='o'
plt.xlabel('Average distance per driven day (scaled)')
plt.ylabel('Average number of speed-limit crossing (scaled)')
plt.show()
```



In [9]:
```
distortions = []
for i in range(1, 11):
    km = KMeans(n_clusters=i, n_init=10, random_state=10)
    km.fit(dfX_scaled)
    distortions.append(km.inertia_)
plt.plot(range(1, 11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```



In [10]:
```
kmeans = KMeans(n_clusters=4, n_init=10, random_state=10) #4, 5
cls = kmeans.fit_predict(dfX_scaled) #6, 7
print(cls)
```

```
[3 3 3 ... 1 1 1]
```

In [11]:
```
print(kmeans.cluster_centers_[2]) #8
```

```
[1.90400473 4.34582367]
```

In [12]:
```
df['clstr_num'] = cls #9
display(df.head())
```

	driver_id	avg_dist_day	avg_over_speed	clstr_num
0	3423311935	71.24	28	3
1	3423313212	52.53	25	3
2	3423313724	64.54	27	3
3	3423311373	55.69	22	3
4	3423310999	54.58	25	3

In [13]:
```
print(df['clstr_num'].value_counts()) #10
```

```
clstr_num
0    2774
1     695
3     427
2     104
Name: count, dtype: int64
```

In [14]:
```
plt.scatter(df['avg_dist_day'], df['avg_over_speed'],
            marker='o', c=df['clstr_num'])
plt.colorbar()
plt.xlabel('avg_dist_day')
plt.ylabel('avg_over_speed')
plt.show()
```