Assignment 8

July 17, 2023

Review Chapter 5 in this week's reading assignment and apply K-means clustering to the used car dataset. Turning in an Ipython notebook and experiment with different cluster sizes. Describe any insights on the used car market from the clustering experiments.

DATA EXPLORATION AND ANALYSIS

[1]:
```
# Imports
import pandas as pd
import numpy as np
```

[2]:
```
file= r"C:\Users\annet\Downloads\CARS\cars.csv"
df= pd.read_csv(file)
df.head()
```

[2]:

	manufacturer_name	model_name	transmission	color	odometer_value	\
0	Subaru	Outback	automatic	silver	190000	
1	Subaru	Outback	automatic	blue	290000	
2	Subaru	Forester	automatic	red	402000	
3	Subaru	Impreza	mechanical	blue	10000	
4	Subaru	Legacy	automatic	black	280000	

	year_produced	engine_fuel	engine_has_gas	engine_type	engine_capacity	\
0	2010	gasoline	False	gasoline	2.5	
1	2002	gasoline	False	gasoline	3.0	
2	2001	gasoline	False	gasoline	2.5	
3	1999	gasoline	False	gasoline	3.0	
4	2001	gasoline	False	gasoline	2.5	

	…	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	\
0	…	True	True	True	False	True	False	
1	…	True	False	False	True	True	False	
2	…	True	False	False	False	False	False	
3	…	False	False	False	False	False	False	
4	…	True	False	True	True	False	False	

	feature_7	feature_8	feature_9	duration_listed
0	True	True	True	16
1	False	False	True	83
2	False	True	True	151

```
3     False      False      False                86
4     False      False       True                 7

[5 rows x 30 columns]
```

[3]:
```
#What types of columns do we have?
df.dtypes.unique()
```

[3]:
```
array([dtype('O'), dtype('int64'), dtype('bool'), dtype('float64')],
      dtype=object)
```

[4]:
```
df.describe()
```

[4]:

	odometer_value	year_produced	engine_capacity	price_usd	\
count	38531.000000	38531.000000	38521.000000	38531.000000	
mean	248864.638447	2002.943734	2.055161	6639.971021	
std	136072.376530	8.065731	0.671178	6428.152018	
min	0.000000	1942.000000	0.200000	1.000000	
25%	158000.000000	1998.000000	1.600000	2100.000000	
50%	250000.000000	2003.000000	2.000000	4800.000000	
75%	325000.000000	2009.000000	2.300000	8990.000000	
max	1000000.000000	2019.000000	8.000000	50000.000000	

	number_of_photos	up_counter	duration_listed
count	38531.000000	38531.000000	38531.000000
mean	9.649062	16.306091	80.577249
std	6.093217	43.286933	112.826569
min	1.000000	1.000000	0.000000
25%	5.000000	2.000000	23.000000
50%	8.000000	5.000000	59.000000
75%	12.000000	16.000000	91.000000
max	86.000000	1861.000000	2232.000000

[5]:
```
corr= df.corr()
corr
```

[5]:

	odometer_value	year_produced	engine_has_gas	\
odometer_value	1.000000	-0.488679	0.057786	
year_produced	-0.488679	1.000000	-0.074686	
engine_has_gas	0.057786	-0.074686	1.000000	
engine_capacity	0.105704	0.005059	0.084579	
has_warranty	-0.189498	0.209231	-0.020667	
price_usd	-0.421204	0.705511	-0.062528	
is_exchangeable	0.042342	-0.057937	0.018646	
number_of_photos	-0.143708	0.258180	-0.032101	
up_counter	-0.020961	0.007945	0.000058	
feature_0	0.105917	-0.346980	0.004089	

feature_1	-0.150960	0.424750	-0.008071
feature_2	-0.076169	0.205039	-0.012030
feature_3	-0.229082	0.431307	-0.027034
feature_4	-0.067292	0.184757	0.003547
feature_5	-0.240968	0.435241	-0.031172
feature_6	-0.166413	0.345745	-0.030140
feature_7	-0.254539	0.457091	-0.036635
feature_8	-0.208844	0.465969	-0.041992
feature_9	-0.087875	0.247439	-0.005855
duration_listed	-0.000428	-0.017001	0.018264

	engine_capacity	has_warranty	price_usd	is_exchangeable	\
odometer_value	0.105704	-0.189498	-0.421204	0.042342	
year_produced	0.005059	0.209231	0.705511	-0.057937	
engine_has_gas	0.084579	-0.020667	-0.062528	0.018646	
engine_capacity	1.000000	-0.054583	0.296597	0.081636	
has_warranty	-0.054583	1.000000	0.285532	0.117775	
price_usd	0.296597	0.285532	1.000000	-0.000503	
is_exchangeable	0.081636	0.117775	-0.000503	1.000000	
number_of_photos	0.106691	0.084045	0.316859	0.103725	
up_counter	0.079152	-0.023087	0.057382	0.106213	
feature_0	-0.142097	0.149966	-0.223896	0.013844	
feature_1	0.134315	-0.121138	0.255806	-0.037863	
feature_2	0.375726	-0.046124	0.338166	0.045769	
feature_3	0.243494	-0.036166	0.470929	-0.029301	
feature_4	0.452533	-0.046572	0.336143	0.010207	
feature_5	0.273235	-0.044887	0.434471	-0.022763	
feature_6	0.284419	-0.021049	0.451714	0.000588	
feature_7	0.202070	-0.024925	0.498547	-0.021710	
feature_8	0.240077	-0.058637	0.449131	0.000701	
feature_9	0.245828	-0.094622	0.266156	0.031275	
duration_listed	0.080081	-0.061798	0.033524	0.026897	

	number_of_photos	up_counter	feature_0	feature_1	\
odometer_value	-0.143708	-0.020961	0.105917	-0.150960	
year_produced	0.258180	0.007945	-0.346980	0.424750	
engine_has_gas	-0.032101	0.000058	0.004089	-0.008071	
engine_capacity	0.106691	0.079152	-0.142097	0.134315	
has_warranty	0.084045	-0.023087	0.149966	-0.121138	
price_usd	0.316859	0.057382	-0.223896	0.255806	
is_exchangeable	0.103725	0.106213	0.013844	-0.037863	
number_of_photos	1.000000	0.073891	-0.106468	0.087605	
up_counter	0.073891	1.000000	-0.022423	0.043414	
feature_0	-0.106468	-0.022423	1.000000	-0.676718	
feature_1	0.087605	0.043414	-0.676718	1.000000	
feature_2	0.131281	0.040796	-0.292278	0.253534	
feature_3	0.177874	0.032885	-0.335893	0.325128	

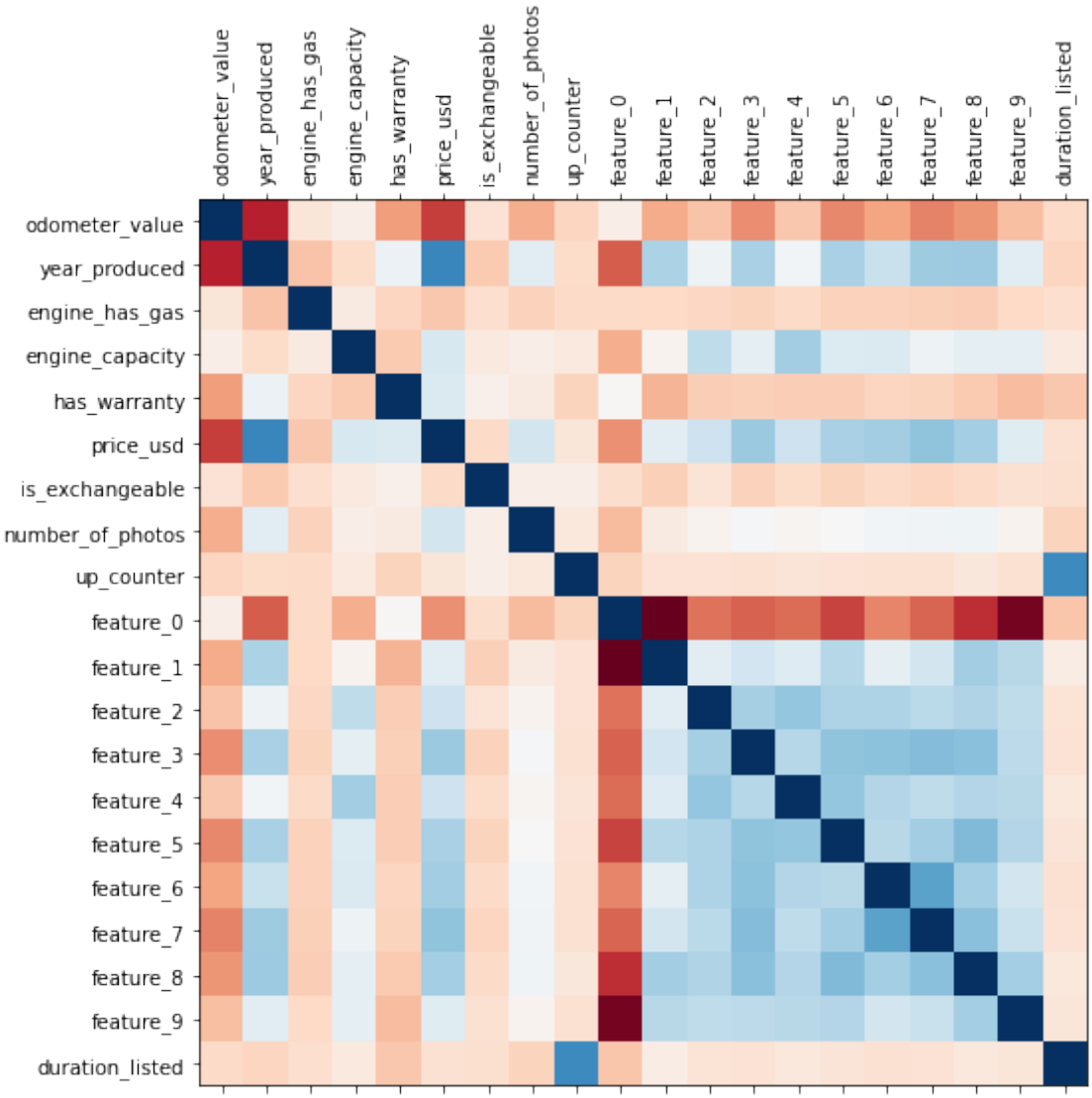
feature_4	0.137266	0.050971	-0.307094	0.267914
feature_5	0.162720	0.039292	-0.404747	0.399224
feature_6	0.183741	0.033456	-0.247182	0.242606
feature_7	0.192636	0.035347	-0.325759	0.316886
feature_8	0.197508	0.063588	-0.458741	0.454120
feature_9	0.132390	0.037183	-0.638831	0.396397
duration_listed	-0.028255	0.698116	-0.068888	0.099893

	feature_2	feature_3	feature_4	feature_5	feature_6	\
odometer_value	-0.076169	-0.229082	-0.067292	-0.240968	-0.166413	
year_produced	0.205039	0.431307	0.184757	0.435241	0.345745	
engine_has_gas	-0.012030	-0.027034	0.003547	-0.031172	-0.030140	
engine_capacity	0.375726	0.243494	0.452533	0.273235	0.284419	
has_warranty	-0.046124	-0.036166	-0.046572	-0.044887	-0.021049	
price_usd	0.338166	0.470929	0.336143	0.434471	0.451714	
is_exchangeable	0.045769	-0.029301	0.010207	-0.022763	0.000588	
number_of_photos	0.131281	0.177874	0.137266	0.162720	0.183741	
up_counter	0.040796	0.032885	0.050971	0.039292	0.033456	
feature_0	-0.292278	-0.335893	-0.307094	-0.404747	-0.247182	
feature_1	0.253534	0.325128	0.267914	0.399224	0.242606	
feature_2	1.000000	0.437269	0.484722	0.422779	0.419173	
feature_3	0.437269	1.000000	0.399051	0.496255	0.502680	
feature_4	0.484722	0.399051	1.000000	0.487586	0.405451	
feature_5	0.422779	0.496255	0.487586	1.000000	0.392463	
feature_6	0.419173	0.502680	0.405451	0.392463	1.000000	
feature_7	0.387818	0.522689	0.372144	0.451348	0.607673	
feature_8	0.412211	0.508890	0.405338	0.534623	0.445521	
feature_9	0.373766	0.378395	0.395459	0.409265	0.312640	
duration_listed	0.049881	0.038335	0.071438	0.051403	0.036499	

	feature_7	feature_8	feature_9	duration_listed
odometer_value	-0.254539	-0.208844	-0.087875	-0.000428
year_produced	0.457091	0.465969	0.247439	-0.017001
engine_has_gas	-0.036635	-0.041992	-0.005855	0.018264
engine_capacity	0.202070	0.240077	0.245828	0.080081
has_warranty	-0.024925	-0.058637	-0.094622	-0.061798
price_usd	0.498547	0.449131	0.266156	0.033524
is_exchangeable	-0.021710	0.000701	0.031275	0.026897
number_of_photos	0.192636	0.197508	0.132390	-0.028255
up_counter	0.035347	0.063588	0.037183	0.698116
feature_0	-0.325759	-0.458741	-0.638831	-0.068888
feature_1	0.316886	0.454120	0.396397	0.099893
feature_2	0.387818	0.412211	0.373766	0.049881
feature_3	0.522689	0.508890	0.378395	0.038335
feature_4	0.372144	0.405338	0.395459	0.071438
feature_5	0.451348	0.534623	0.409265	0.051403
feature_6	0.607673	0.445521	0.312640	0.036499

```
feature_7          1.000000   0.514342   0.351370          0.038122
feature_8          0.514342   1.000000   0.447748          0.070011
feature_9          0.351370   0.447748   1.000000          0.058277
duration_listed    0.038122   0.070011   0.058277          1.000000
```

[6]:
```
import matplotlib.pyplot as plt

fig= plt.figure(figsize=(8,8))
plt.matshow(corr,cmap='RdBu', fignum=fig.number)
plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical');
plt.yticks(range(len(corr.columns)), corr.columns);
```



[7]:
```
df.columns
```

[7]:
```
Index(['manufacturer_name', 'model_name', 'transmission', 'color',
       'odometer_value', 'year_produced', 'engine_fuel', 'engine_has_gas',
       'engine_type', 'engine_capacity', 'body_type', 'has_warranty', 'state',
       'drivetrain', 'price_usd', 'is_exchangeable', 'location_region',
       'number_of_photos', 'up_counter', 'feature_0', 'feature_1', 'feature_2',
       'feature_3', 'feature_4', 'feature_5', 'feature_6', 'feature_7',
       'feature_8', 'feature_9', 'duration_listed'],
      dtype='object')
```

[8]:
```
# Find Categorical Columns
categorical=[]
for col in df.columns:
    if df[col].dtype == "object" or df[col].dtype == "bool":
        categorical.append(col)
```

[9]:
```
categorical
```

[9]:
```
['manufacturer_name',
 'model_name',
 'transmission',
 'color',
 'engine_fuel',
 'engine_has_gas',
 'engine_type',
 'body_type',
 'has_warranty',
 'state',
 'drivetrain',
 'is_exchangeable',
 'location_region',
 'feature_0',
 'feature_1',
 'feature_2',
 'feature_3',
 'feature_4',
 'feature_5',
 'feature_6',
 'feature_7',
 'feature_8',
 'feature_9']
```

[10]:
```
# Find NAN values
df.isna().sum()
```

[10]:
```
manufacturer_name     0
model_name            0
transmission          0
color                 0
odometer_value        0
year_produced         0
engine_fuel           0
engine_has_gas        0
engine_type           0
engine_capacity      10
body_type             0
has_warranty          0
state                 0
drivetrain            0
price_usd             0
is_exchangeable       0
location_region       0
number_of_photos      0
up_counter            0
feature_0             0
feature_1             0
feature_2             0
feature_3             0
feature_4             0
feature_5             0
feature_6             0
feature_7             0
feature_8             0
feature_9             0
duration_listed       0
dtype: int64
```

[11]:
```
#Since not many, drop NAs
df = df.dropna()
```

[12]:
```
df
```

[12]:

	manufacturer_name	model_name	transmission	color	odometer_value	\
0	Subaru	Outback	automatic	silver	190000	
1	Subaru	Outback	automatic	blue	290000	
2	Subaru	Forester	automatic	red	402000	
3	Subaru	Impreza	mechanical	blue	10000	
4	Subaru	Legacy	automatic	black	280000	
…	…	…	…	…	…	
38526	Chrysler	300	automatic	silver	290000	
38527	Chrysler	PT Cruiser	mechanical	blue	321000	
38528	Chrysler	300	automatic	blue	777957	

```
38529          Chrysler  PT Cruiser   mechanical   black           20000
38530          Chrysler     Voyager    automatic  silver          297729

       year_produced engine_fuel  engine_has_gas engine_type  engine_capacity  \
0               2010    gasoline           False    gasoline              2.5
1               2002    gasoline           False    gasoline              3.0
2               2001    gasoline           False    gasoline              2.5
3               1999    gasoline           False    gasoline              3.0
4               2001    gasoline           False    gasoline              2.5
…                …          …              …          …                …
38526           2000    gasoline           False    gasoline              3.5
38527           2004      diesel           False      diesel              2.2
38528           2000    gasoline           False    gasoline              3.5
38529           2001    gasoline           False    gasoline              2.0
38530           2000    gasoline           False    gasoline              2.4

       … feature_1  feature_2 feature_3 feature_4  feature_5  feature_6  \
0      …      True       True      True     False       True      False
1      …      True      False     False      True       True      False
2      …      True      False     False     False      False      False
3      …     False      False     False     False      False      False
4      …      True      False      True      True      False      False
…    …        …         …        …        …         …         …
38526  …      True      False     False      True       True      False
38527  …      True      False     False      True       True      False
38528  …      True      False     False      True       True      False
38529  …      True      False     False     False      False      False
38530  …     False      False     False     False      False      False

      feature_7  feature_8  feature_9  duration_listed
0          True       True       True               16
1         False      False       True               83
2         False       True       True              151
3         False      False      False               86
4         False      False       True                7
…         …        …          …                …
38526     False       True       True              301
38527     False       True       True              317
38528     False       True       True              369
38529     False      False       True              490
38530     False      False       True              632

[38521 rows x 30 columns]
```

ONE HOT ENCODE: no ordinal categorical data

[13]:
```
one_hot_df= pd.get_dummies(df, columns=categorical)
one_hot_df.head()
```

[13]:
```
   odometer_value  year_produced  engine_capacity  price_usd  \
0          190000           2010              2.5   10900.00
1          290000           2002              3.0    5000.00
2          402000           2001              2.5    2800.00
3           10000           1999              3.0    9999.00
4          280000           2001              2.5    2134.11

   number_of_photos  up_counter  duration_listed  manufacturer_name_Acura  \
0                 9          13               16                        0
1                12          54               83                        0
2                 4          72              151                        0
3                 9          42               86                        0
4                14           7                7                        0

   manufacturer_name_Alfa Romeo  manufacturer_name_Audi  …  feature_5_False  \
0                             0                       0  …                0
1                             0                       0  …                0
2                             0                       0  …                1
3                             0                       0  …                1
4                             0                       0  …                1

   feature_5_True  feature_6_False  feature_6_True  feature_7_False  \
0               1                1               0                0
1               1                1               0                1
2               0                1               0                1
3               0                1               0                1
4               0                1               0                1

   feature_7_True  feature_8_False  feature_8_True  feature_9_False  \
0               1                0               1                0
1               0                1               0                0
2               0                0               1                0
3               0                1               0                1
4               0                1               0                0

   feature_9_True
0               1
1               1
2               1
3               0
4               1

[5 rows x 1249 columns]
```

[14]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_df1 = scaler.fit_transform(one_hot_df)
```

```
scaled_df1
```

```
C:\Users\annet\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

[14]:
```
array([[-0.43297851,  0.87531768,  0.66278228, …,  1.18672453,
        -0.85215057,  0.85215057],
       [ 0.3020036 , -0.11666503,  1.40775119, …, -0.84265554,
        -0.85215057,  0.85215057],
       [ 1.12518356, -0.24066287,  0.66278228, …,  1.18672453,
        -0.85215057,  0.85215057],
       …,
       [ 3.88840025, -0.36466071,  2.15272009, …,  1.18672453,
        -0.85215057,  0.85215057],
       [-1.68244809, -0.24066287, -0.08218662, …, -0.84265554,
        -0.85215057,  0.85215057],
       [ 0.35881037, -0.36466071,  0.5137885 , …, -0.84265554,
        -0.85215057,  0.85215057]])
```

LABEL ENCODING

[15]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
label_df=df.copy()
for column in categorical:
    label_df[column] = label_encoder.fit_transform(label_df[column])
```

[16]:
```
label_df
```

[16]:

	manufacturer_name	model_name	transmission	color	odometer_value	\
0	45	763	0	8	190000	
1	45	763	0	1	290000	
2	45	519	0	7	402000	
3	45	609	1	1	10000	
4	45	664	0	0	280000	
…	…	…	…	…	…	
38526	8	86	0	8	290000	
38527	8	765	1	1	321000	
38528	8	86	0	1	777957	
38529	8	765	1	0	20000	
38530	8	1057	0	8	297729	

	year_produced	engine_fuel	engine_has_gas	engine_type	\
0	2010	2	0	1	
1	2002	2	0	1	
2	2001	2	0	1	
3	1999	2	0	1	

```
4               2001            2               0            1
…               …          …               …         …
38526           2000            2               0            1
38527           2004            0               0            0
38528           2000            2               0            1
38529           2001            2               0            1
38530           2000            2               0            1

       engine_capacity  …  feature_1  feature_2  feature_3  feature_4  \
0                  2.5  …          1          1          1          0
1                  3.0  …          1          0          0          1
2                  2.5  …          1          0          0          0
3                  3.0  …          0          0          0          0
4                  2.5  …          1          0          1          1
…                  …    …       …          …          …       …
38526              3.5  …          1          0          0          1
38527              2.2  …          1          0          0          1
38528              3.5  …          1          0          0          1
38529              2.0  …          1          0          0          0
38530              2.4  …          0          0          0          0

       feature_5  feature_6  feature_7  feature_8  feature_9  duration_listed
0              1          0          1          1          1               16
1              1          0          0          0          1               83
2              0          0          0          1          1              151
3              0          0          0          0          0               86
4              0          0          0          0          1                7
…          …          …          …          …          …                …
38526          1          0          0          1          1              301
38527          1          0          0          1          1              317
38528          1          0          0          1          1              369
38529          0          0          0          0          1              490
38530          0          0          0          0          1              632

[38521 rows x 30 columns]
```

[17]:
```
scaler = StandardScaler()
scaled_df2 = scaler.fit_transform(label_df)
scaled_df2
```

[17]:
```
array([[ 1.08929656,  0.59731369, -1.41028522, …,  1.18672453,
         0.85215057, -0.57237034],
       [ 1.08929656,  0.59731369, -1.41028522, …, -0.84265554,
         0.85215057,  0.02140593],
       [ 1.08929656, -0.14995951, -1.41028522, …,  1.18672453,
         0.85215057,  0.62404454],
       …,
```

```
[-1.23791351, -1.47606318, -1.41028522, …,  1.18672453,
  0.85215057,  2.55603302],
[-1.23791351,  0.60343888,  0.70907642, …, -0.84265554,
  0.85215057,  3.62837525],
[-1.23791351,  1.49771665, -1.41028522, …, -0.84265554,
  0.85215057,  4.88682646]])
```

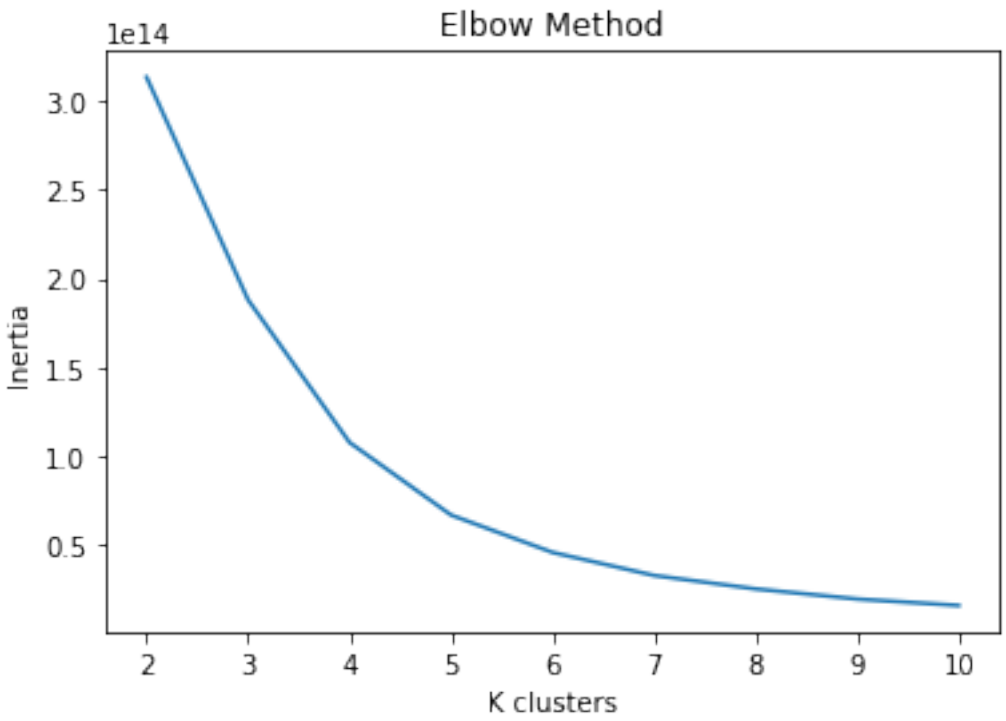
1 KMEANS Clustering

Inertia : sum of squared distances between data points and their assigned cluster centroids. the lower the inertia, the more compact the centroids

[18]:
```
from sklearn.cluster import KMeans
from sklearn import metrics
import matplotlib.pyplot as plt
```

One Hot Kmeans

[19]:
```
clusters = range(2, 11)
inertia = []
matrix= one_hot_df

for k in clusters:
    kmeans = KMeans(n_clusters=k, random_state=2023)
    kmeans.fit(matrix)
    inertia.append(kmeans.inertia_)

plt.plot(clusters, inertia)
plt.xlabel('K clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```

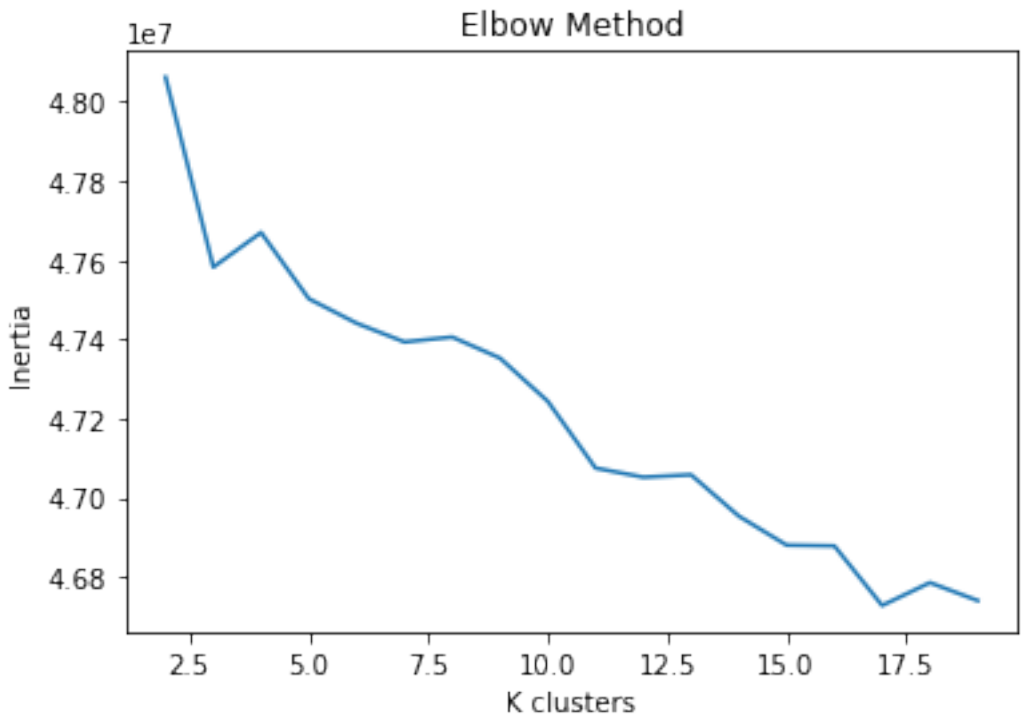


[20]:
```
print("The optimal number of clusters is 5 with an inertia of",inertia[3])
```

```
The optimal number of clusters is 5 with an inertia of 67022369333960.67
```

One Hot Kmeans with Standard Scaling

[21]:
```
clusters = range(2, 20)
inertia = []
matrix= scaled_df1

for k in clusters:
    kmeans = KMeans(n_clusters=k, random_state=2023)
    kmeans.fit(matrix)
    inertia.append(kmeans.inertia_)

plt.plot(clusters, inertia)
plt.xlabel('K clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```

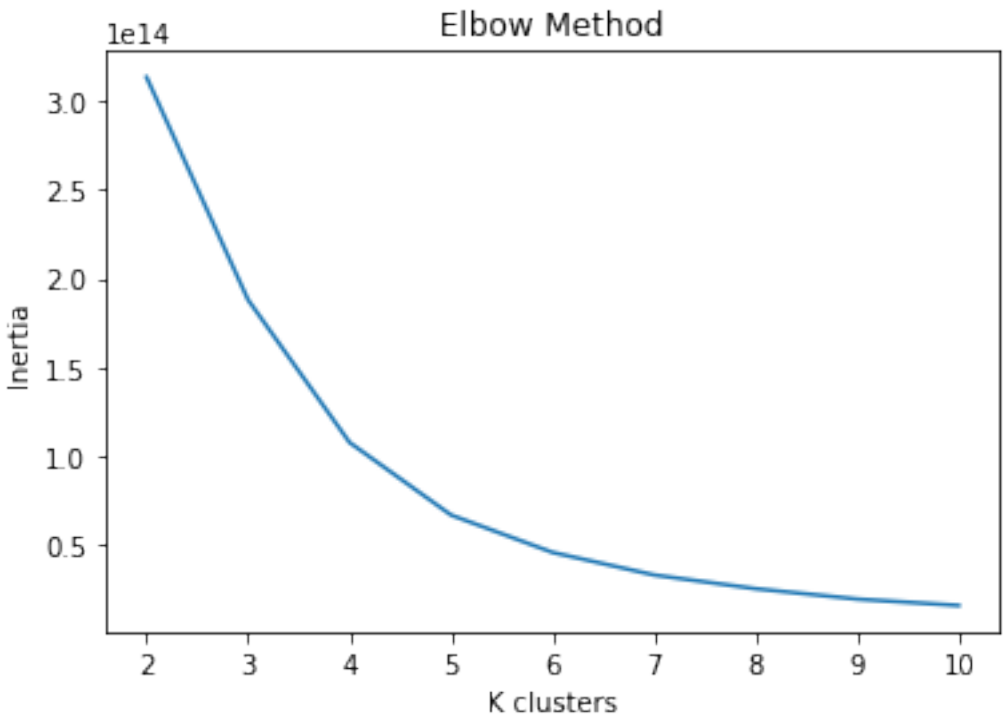


[23]:
```
print("The optimal number of clusters is 5 with an inertia of",inertia[16])
```

```
The optimal number of clusters is 5 with an inertia of 46785026.73741194
```

Label Encoding

[24]:
```
clusters = range(2, 11)
inertia = []
matrix= label_df

for k in clusters:
    kmeans = KMeans(n_clusters=k, random_state=2023)
    kmeans.fit(matrix)
    inertia.append(kmeans.inertia_)

plt.plot(clusters, inertia)
plt.xlabel('K clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```



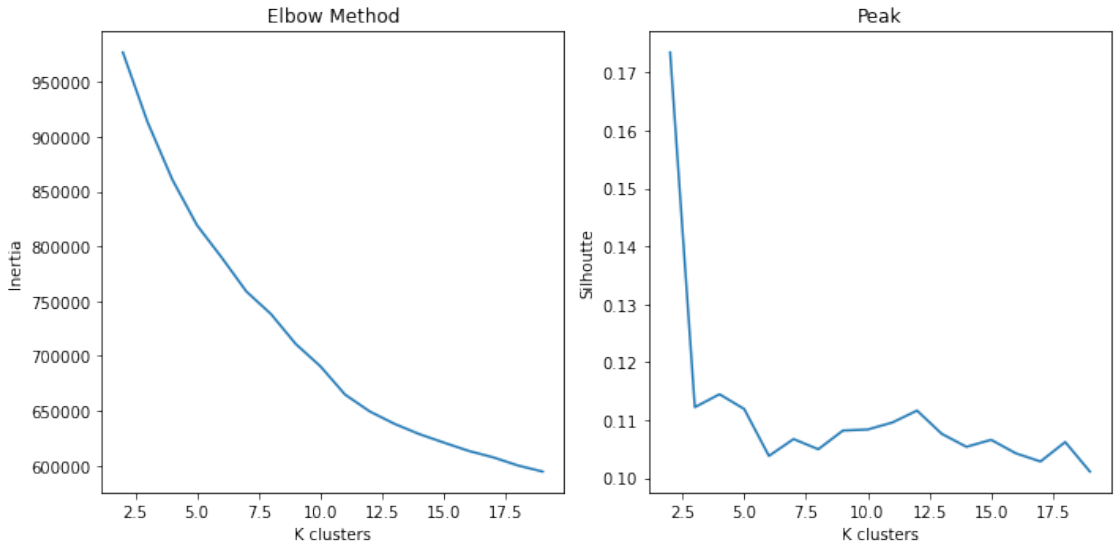
[25]:
```
print("The optimal number of clusters is 6 with an inertia of",inertia[4])
```

```
The optimal number of clusters is 6 with an inertia of 46008374545481.5
```

Labled Scaled

[26]:
```
clusters = range(2, 20)
inertia = []
silhouette=[]
matrix= scaled_df2

for k in clusters:
    kmeans = KMeans(n_clusters=k, random_state=2023)
    kmeans.fit(matrix)
    inertia.append(kmeans.inertia_)
    silhouette.append(metrics.silhouette_score(matrix, kmeans.labels_))

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(clusters, inertia)
plt.xlabel('K clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')

plt.subplot(1, 2, 2)
```

```
plt.plot(clusters, silhouette)
plt.xlabel('K clusters')
plt.ylabel('Silhoutte')
plt.title('Peak')

plt.tight_layout()
plt.show()
```



[27]:
```
print("The optimal number of clusters is 12 with an inertia of",inertia[10])
```

```
The optimal number of clusters is 12 with an inertia of 649366.1303156037
```

1.1 Further Analysis

[28]:
```
kmeans = KMeans(n_clusters=12, random_state=2023)
kmeans.fit(matrix)
```

[28]:
```
KMeans(n_clusters=12, random_state=2023)
```

[29]:
```
cluster_labels = kmeans.labels_
```

[30]:
```
df.columns
```

[30]:
```
Index(['manufacturer_name', 'model_name', 'transmission', 'color',
       'odometer_value', 'year_produced', 'engine_fuel', 'engine_has_gas',
       'engine_type', 'engine_capacity', 'body_type', 'has_warranty', 'state',
       'drivetrain', 'price_usd', 'is_exchangeable', 'location_region',
       'number_of_photos', 'up_counter', 'feature_0', 'feature_1', 'feature_2',
       'feature_3', 'feature_4', 'feature_5', 'feature_6', 'feature_7',
```

```
       'feature_8', 'feature_9', 'duration_listed'],
      dtype='object')
```

[31]:
```
new_df=scaled_df2.copy()
new_df = pd.DataFrame(new_df)
new_df.columns=df.columns
new_df["cluster"]= cluster_labels
cluster_stats = new_df.groupby('cluster').mean()
```

[32]:
```
new_df
```

[32]:

	manufacturer_name	model_name	transmission	color	odometer_value	\
0	1.089297	0.597314	-1.410285	0.978100	-0.432979	
1	1.089297	0.597314	-1.410285	-0.969685	0.302004	
2	1.089297	-0.149960	-1.410285	0.699845	1.125184	
3	1.089297	0.125674	0.709076	-0.969685	-1.755946	
4	1.089297	0.294117	-1.410285	-1.247940	0.228505	
…	…	…	…	…	…	
38516	-1.237914	-1.476063	-1.410285	0.978100	0.302004	
38517	-1.237914	0.603439	0.709076	-0.969685	0.529848	
38518	-1.237914	-1.476063	-1.410285	-0.969685	3.888400	
38519	-1.237914	0.603439	0.709076	-1.247940	-1.682448	
38520	-1.237914	1.497717	-1.410285	0.978100	0.358810	

	year_produced	engine_fuel	engine_has_gas	engine_type	\
0	0.875318	0.721021	-0.190355	0.708498	
1	-0.116665	0.721021	-0.190355	0.708498	
2	-0.240663	0.721021	-0.190355	0.708498	
3	-0.488659	0.721021	-0.190355	0.708498	
4	-0.240663	0.721021	-0.190355	0.708498	
…	…	…	…	…	
38516	-0.364661	0.721021	-0.190355	0.708498	
38517	0.131331	-1.365794	-0.190355	-1.411437	
38518	-0.364661	0.721021	-0.190355	0.708498	
38519	-0.240663	0.721021	-0.190355	0.708498	
38520	-0.364661	0.721021	-0.190355	0.708498	

	engine_capacity	…	feature_2	feature_3	feature_4	feature_5	\
0	0.662782	…	1.862331	1.620868	-0.564189	1.345099	
1	1.407751	…	-0.536962	-0.616953	1.772455	1.345099	
2	0.662782	…	-0.536962	-0.616953	-0.564189	-0.743440	
3	1.407751	…	-0.536962	-0.616953	-0.564189	-0.743440	
4	0.662782	…	-0.536962	1.620868	1.772455	-0.743440	
…	…	…	…	…	…	…	
38516	2.152720	…	-0.536962	-0.616953	1.772455	1.345099	
38517	0.215801	…	-0.536962	-0.616953	1.772455	1.345099	
38518	2.152720	…	-0.536962	-0.616953	1.772455	1.345099	

```
38519        -0.082187  …  -0.536962  -0.616953  -0.564189  -0.743440
38520         0.513789  …  -0.536962  -0.616953  -0.564189  -0.743440

       feature_6  feature_7  feature_8  feature_9  duration_listed  cluster
0      -0.454002   1.671653   1.186725   0.852151        -0.572370        8
1      -0.454002  -0.598210  -0.842656   0.852151         0.021406        3
2      -0.454002  -0.598210   1.186725   0.852151         0.624045       11
3      -0.454002  -0.598210  -0.842656  -1.173502         0.047993        1
4      -0.454002  -0.598210  -0.842656   0.852151        -0.652131        3
…            …          …          …          …                …        …
38516  -0.454002  -0.598210   1.186725   0.852151         1.953394        3
38517  -0.454002  -0.598210   1.186725   0.852151         2.095192        4
38518  -0.454002  -0.598210   1.186725   0.852151         2.556033        3
38519  -0.454002  -0.598210  -0.842656   0.852151         3.628375       11
38520  -0.454002  -0.598210  -0.842656   0.852151         4.886826       11

[38521 rows x 31 columns]
```

[33]:
```
cluster_stats
```

[33]:
```
         manufacturer_name  model_name  transmission     color  \
cluster
0                 0.017645   -0.052814      0.305937 -0.028306
1                 0.173837   -0.237452      0.476106  0.075449
2                -0.157051    0.165354     -0.619074 -0.301318
3                -0.399301   -0.226787     -0.883499 -0.257393
4                 0.159392    0.153530      0.510140  0.112533
5                -0.146164    0.050920     -0.066820  0.031129
6                -0.026462    0.042448     -0.057852 -0.035311
7                 0.118456    0.288455      0.568181  0.400658
8                -0.377251    0.102771     -1.260418 -0.251091
9                 0.824538    0.567322     -0.603134 -0.095877
10                0.152035    0.063490     -0.374814  0.044078
11               -0.037435   -0.176705      0.440326  0.010741

         odometer_value  year_produced  engine_fuel  engine_has_gas  \
cluster
0              0.137298      -0.726883     0.216330       -0.027636
1              0.057780      -0.833416     0.724735       -0.190355
2             -0.161389       0.773384    -1.365794       -0.190355
3              0.179312      -0.065031     0.476887       -0.190355
4              0.467102      -0.079098    -1.365794       -0.190355
5              0.319060      -0.412456    -0.322386        5.253344
6             -0.224174      -0.235356    -0.058489        0.021462
7              0.683014      -0.584369    -1.365794       -0.190355
8             -0.641119       0.851135     0.763472       -0.130592
9             -1.745686       1.927504     0.465398       -0.190355
```

```
10            -0.830574       1.014300     0.751132       -0.190355
11             0.174966      -0.462907     0.727297       -0.190355

         engine_type  engine_capacity  …  feature_1  feature_2  feature_3  \
cluster                                 …
0           0.218839        -0.209722  …  -0.659016  -0.334847  -0.452765
1           0.708498        -0.476086  …  -1.243248  -0.536962  -0.616953
2          -1.411437         0.386060  …   0.535714   0.966615   1.342622
3           0.454738         0.867706  …   0.535637   0.672854   0.163135
4          -1.411437        -0.054746  …   0.307607  -0.223217  -0.293902
5           0.708498         0.369408  …  -0.051460  -0.107986  -0.178742
6          -0.058638         0.450597  …   0.079322   0.107206   0.018692
7          -1.411437         0.143181  …  -1.243248  -0.526048  -0.613038
8           0.705075         1.246904  …   0.591453   1.449408   1.324611
9           0.448818        -0.502621  …  -1.115558  -0.424745  -0.332865
10          0.693204        -0.516657  …   0.606561  -0.227313   0.441863
11          0.708226        -0.388277  …   0.073343  -0.309159  -0.494682

         feature_4  feature_5  feature_6  feature_7  feature_8  feature_9  \
cluster
0        -0.221312  -0.442645  -0.266305  -0.394663  -0.555895  -0.612044
1        -0.555874  -0.743027  -0.454002  -0.598210  -0.842656  -1.173502
2         0.871001   0.866082   1.629746   1.320811   1.066568   0.719590
3         1.426198   0.795521  -0.304460  -0.217975   0.588026   0.727823
4        -0.287377  -0.346959  -0.304811  -0.290737  -0.051135   0.101096
5        -0.025242  -0.201907  -0.217537  -0.238054  -0.255203  -0.046223
6         0.244999   0.044841   0.135212   0.028873   0.199672   0.008786
7        -0.546202  -0.741247  -0.453073  -0.594239  -0.841945  -1.173502
8         1.466888   1.233827   1.719686   1.522870   1.126439   0.822063
9        -0.428882  -0.413181  -0.193664  -0.229168  -0.539830  -0.871233
10       -0.402638   0.770437  -0.093041   0.500786   0.632407   0.308188
11       -0.402134  -0.533918  -0.420688  -0.478835  -0.523512   0.102283

         duration_listed
cluster
0              -0.015705
1              -0.178603
2              -0.015123
3               0.065579
4              -0.034165
5               0.018086
6               7.726187
7              -0.152814
8               0.089975
9              -0.569133
10             -0.044177
11             -0.051123
```

```
[12 rows x 30 columns]
```

[36]:
```
import seaborn as sns
```

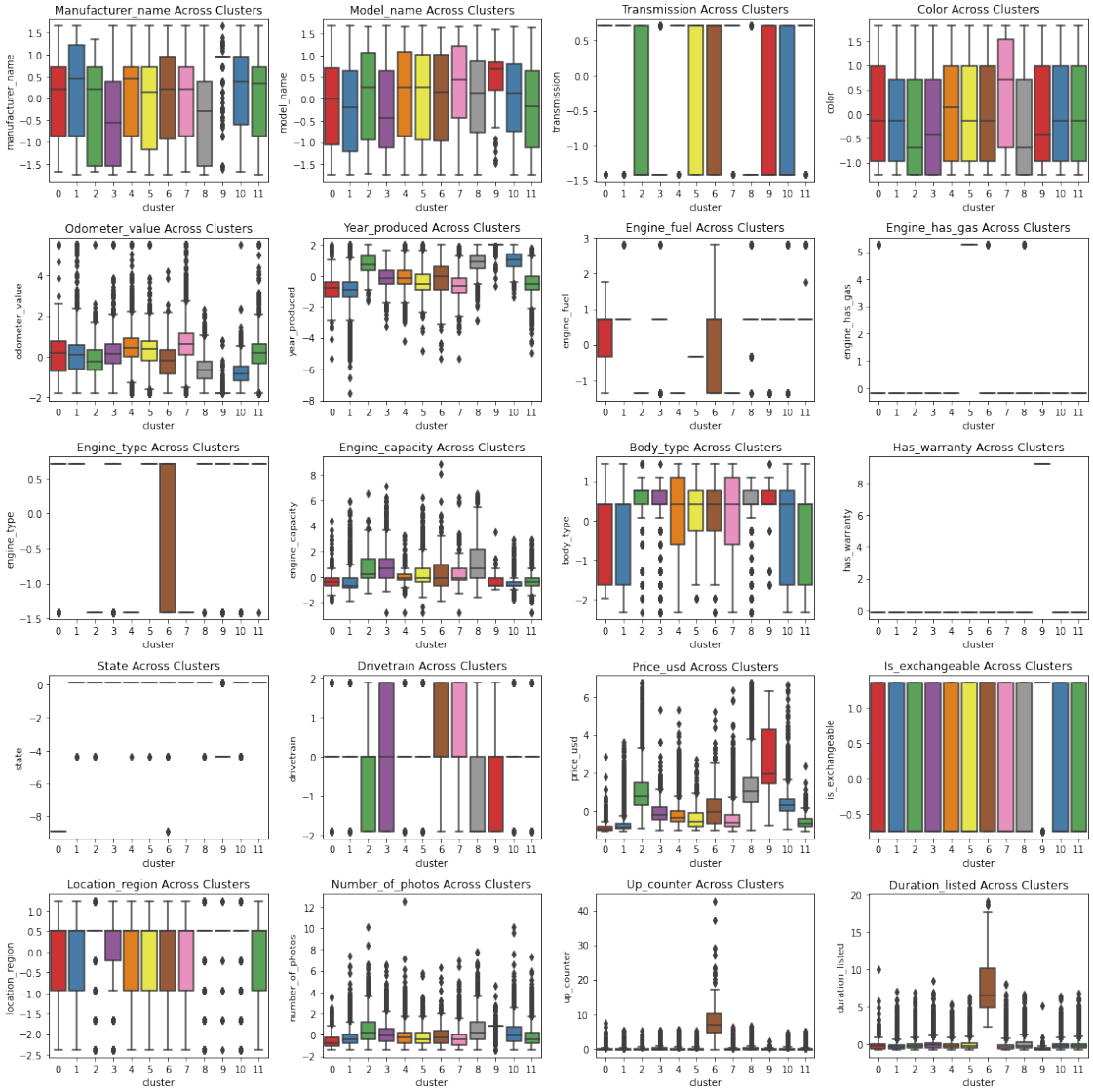
[42]:
```
plt.figure(figsize=(16, 12))

plt.subplot(2, 2, 1)
sns.scatterplot(x='year_produced', y='odometer_value', hue='cluster',␣
 ↪data=new_df, palette='Set1')
plt.title('Year Produced vs. Odometer Value')

plt.subplot(2, 2, 2)
sns.scatterplot(x='year_produced', y='price_usd', hue='cluster', data=new_df,␣
 ↪palette='Set1')
plt.title('Year Produced vs. Price')

plt.subplot(2, 2, 3)
sns.scatterplot(x='engine_capacity', y='odometer_value', hue='cluster',␣
 ↪data=new_df, palette='Set1')
plt.title('Engine Capacity vs. Odometer Value')

plt.subplot(2, 2, 4)
sns.scatterplot(x='engine_capacity', y='price_usd', hue='cluster', data=new_df,␣
 ↪palette='Set1')
plt.title('Engine Capacity vs. Price')

plt.tight_layout()
plt.show()
```

[45]:
```
features=['manufacturer_name', 'model_name', 'transmission',␣
 ↪'color','odometer_value', 'year_produced', 'engine_fuel', 'engine_has_gas',
       'engine_type', 'engine_capacity', 'body_type', 'has_warranty', 'state',
       'drivetrain', 'price_usd', 'is_exchangeable', 'location_region',
       'number_of_photos', 'up_counter','duration_listed']
```

[48]:
```
len(features)
```

[48]: 20

[53]:
```
plt.figure(figsize=(16, 16))
for i, feature in enumerate(features, 1):
    plt.subplot(5, 4, i)
    sns.boxplot(x='cluster', y=feature, data=new_df, palette='Set1')
    plt.title(f'{feature.capitalize()} Across Clusters')

plt.tight_layout()
plt.show()
```



From the analysis, we can note that standardization applied to label encoding produced the lowest inertia. The optimal k was found to be 12. We can see that price, year, and odometer value may have influenced the clusters a lot, as there is fewer overlap across the clusters. Although we cannot verify this numerically, it seems as if both the box plots and the scatter plots have fairly distinguished clusters associated with price. We could analyze further which features may or may not be necessary depending on their correlation relationship. We do seem to see that price, odometer value, and year all seem to be correlated with each other. It is not surprising to thus see a distinguishment in the plots for those three features. It should be noted that a PCA or TSNE plot would really help visualize the clusters, but in past homework assignments, dimensional reduction was found to lose too much information.

[ ]: