21BDS0340 - Abhinav Dinesh Srivatsa

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA, FactorAnalysis
from sklearn.cluster import SpectralClustering, KMeans, AgglomerativeClus
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.manifold import MDS
from minisom import MiniSom
from sklearn.metrics import mean_absolute_error, root_mean_squared_error,
```

In [2]:
```
data = pd.read_csv("NaturalGas.csv")
data
```

Out[2]:

	rownames	state	statecode	year	consumption	price	eprice	oprice	lprice
0	1	NY	35	1967	313656	1.42	2.98	7.40	1.47
1	2	NY	35	1968	319282	1.38	2.91	7.77	1.42
2	3	NY	35	1969	331326	1.37	2.84	7.96	1.38
3	4	NY	35	1970	346533	1.40	2.87	8.33	1.37
4	5	NY	35	1971	352085	1.50	3.07	8.80	1.40
...	...	...	...	...	...	...	...	...	...
133	134	CA	5	1985	527495	5.72	7.78	30.58	5.84
134	135	CA	5	1986	464307	5.14	7.95	44.15	5.72
135	136	CA	5	1987	503473	5.26	8.03	35.24	5.14
136	137	CA	5	1988	497138	5.64	8.69	34.02	5.26
137	138	CA	5	1989	514276	5.59	9.45	44.44	5.64

138 rows × 11 columns

Module 2 - Data Transformations

In [3]:
```
# data deduplication
deduplicated = data.drop_duplicates()
deduplicated
```

Out[3]:

	rownames	state	statecode	year	consumption	price	eprice	oprice	lprice
0	1	NY	35	1967	313656	1.42	2.98	7.40	1.47
1	2	NY	35	1968	319282	1.38	2.91	7.77	1.42
2	3	NY	35	1969	331326	1.37	2.84	7.96	1.38
3	4	NY	35	1970	346533	1.40	2.87	8.33	1.37
4	5	NY	35	1971	352085	1.50	3.07	8.80	1.40
...	...	...	...	...	...	...	...	...	...
133	134	CA	5	1985	527495	5.72	7.78	30.58	5.84
134	135	CA	5	1986	464307	5.14	7.95	44.15	5.72
135	136	CA	5	1987	503473	5.26	8.03	35.24	5.14
136	137	CA	5	1988	497138	5.64	8.69	34.02	5.26
137	138	CA	5	1989	514276	5.59	9.45	44.44	5.64

138 rows × 11 columns

In [4]:

```
# checking missing values
missing = data.isna().sum()
missing
```

Out[4]:

```
rownames       0
state          0
statecode      0
year           0
consumption    0
price          0
eprice         0
oprice         0
lprice         0
heating        0
income         0
dtype: int64
```

In [5]:

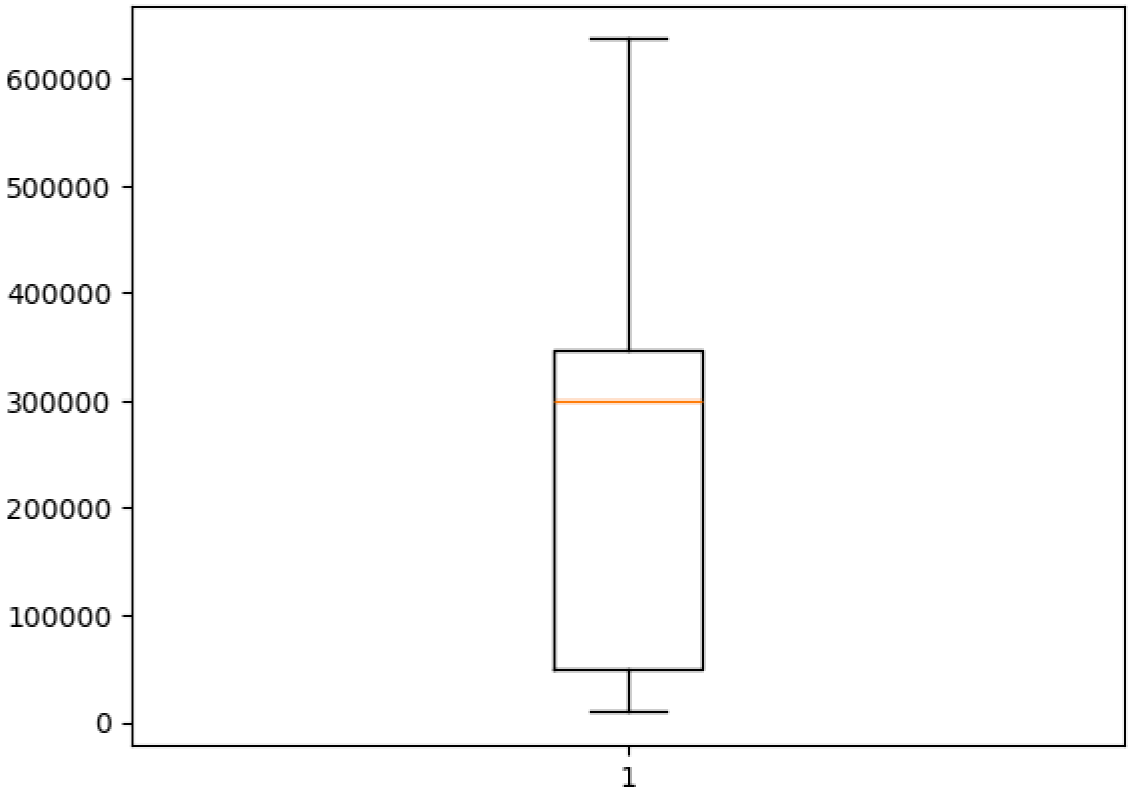
```
# binning the year
pd.qcut(data.year, q=4, labels=["q1", "q2", "q3", "q4"])
```

Out[5]:

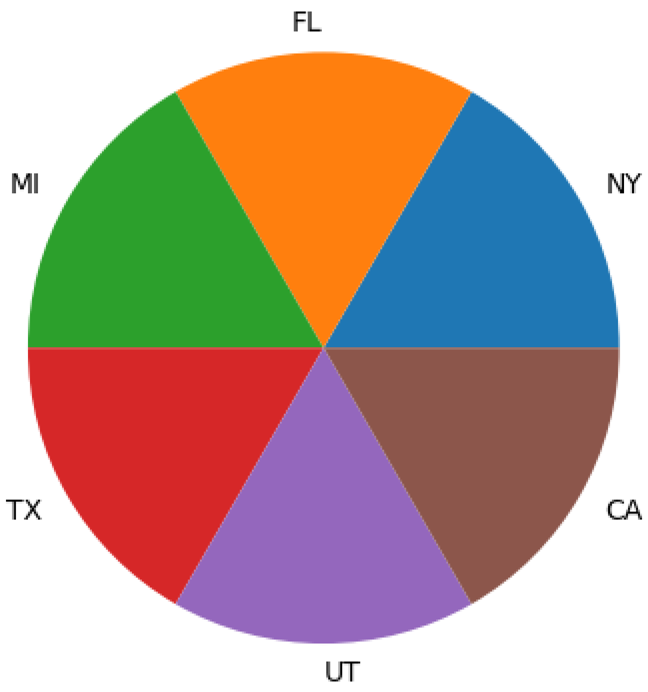
```
0      q1
1      q1
2      q1
3      q1
4      q1
       ..
133    q4
134    q4
135    q4
136    q4
137    q4
Name: year, Length: 138, dtype: category
Categories (4, object): ['q1' < 'q2' < 'q3' < 'q4']
```

Module 3 - Correlation ANalysis and Time Series Analysis

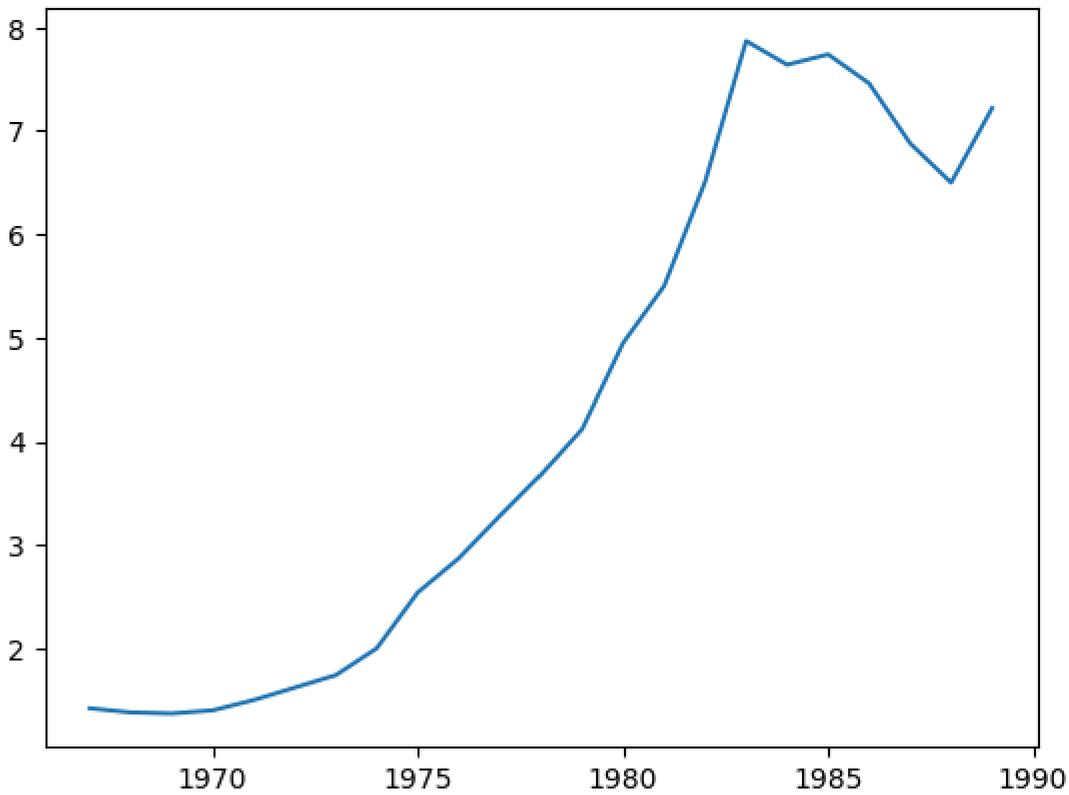
In [6]:
```
# univariate analysis
# checking consumption range
plt.boxplot(data.consumption)
plt.show()
```



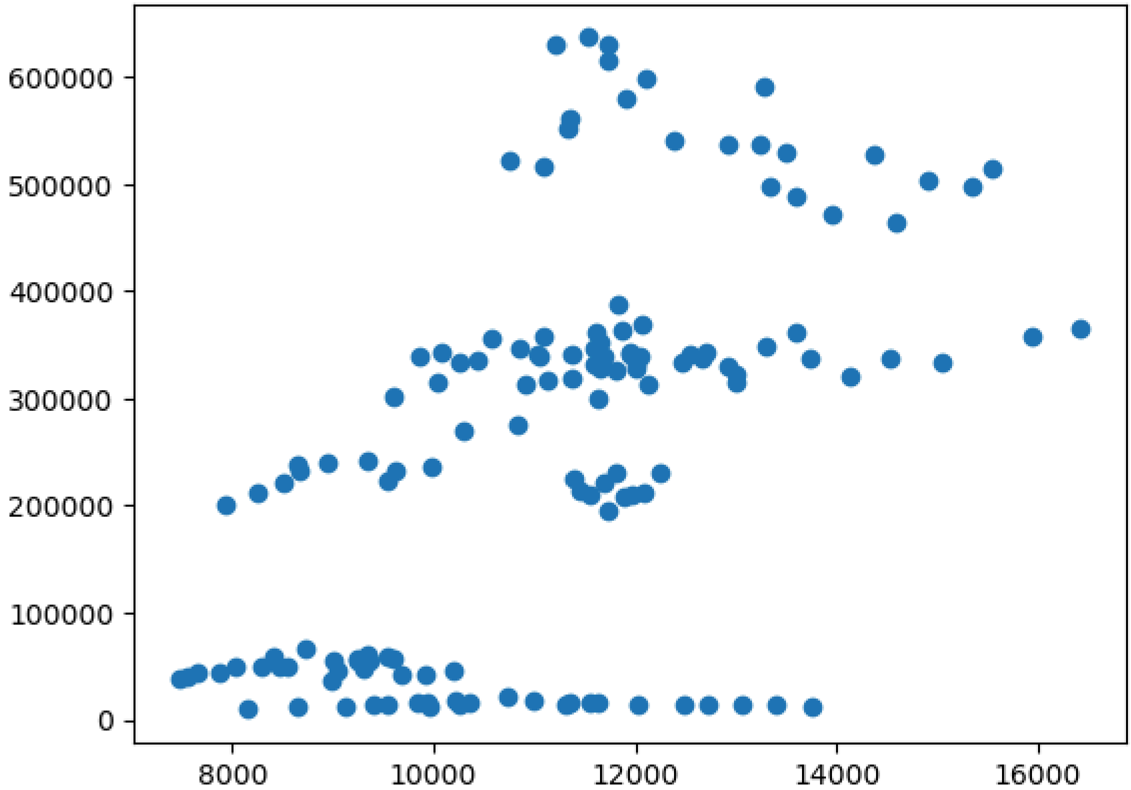
In [7]:
```
# checking pie chart of state
state_counts = data.state.value_counts()
plt.pie(state_counts, labels=state_counts.index)
plt.show()
```



In [8]:
```
# bivariate analysis
# checking price vs. year for state NY
ny_data = data[data.state == "NY"]
plt.plot(ny_data.year, ny_data.price)
plt.show()
```

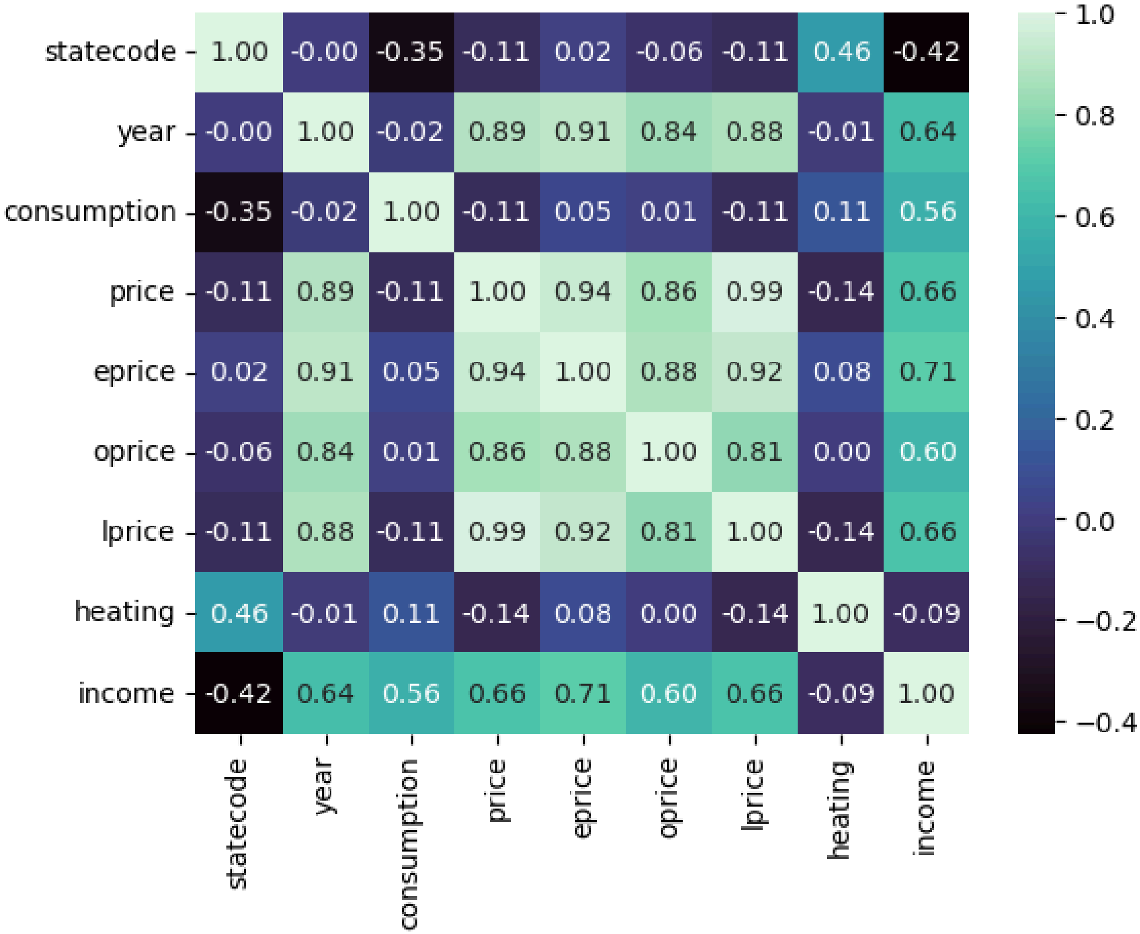


In [9]:
```
# checking income vs. consumption
plt.scatter(data.income, data.consumption)
plt.show()
```



In [10]:
```
# multivariate analysis
numeric_data = data.drop(["rownames", "state"], axis=1)
sns.heatmap(numeric_data.corr(), cmap="mako", annot=True, fmt=".2f")
```

Out[10]: <Axes: >



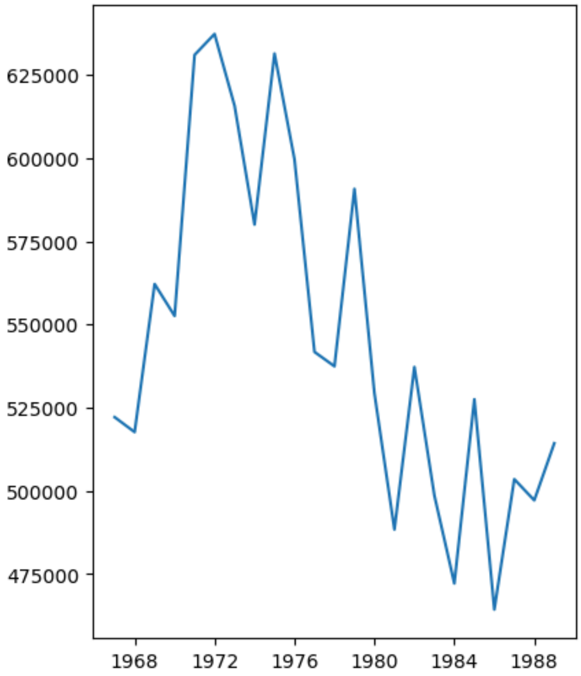
In [11]:
```
# time series analysis
ts_data = data.copy()
ts_data.year = pd.to_datetime(ts_data.year, format="%Y")
ts_data.set_index("year", inplace=True)
ts_data
```

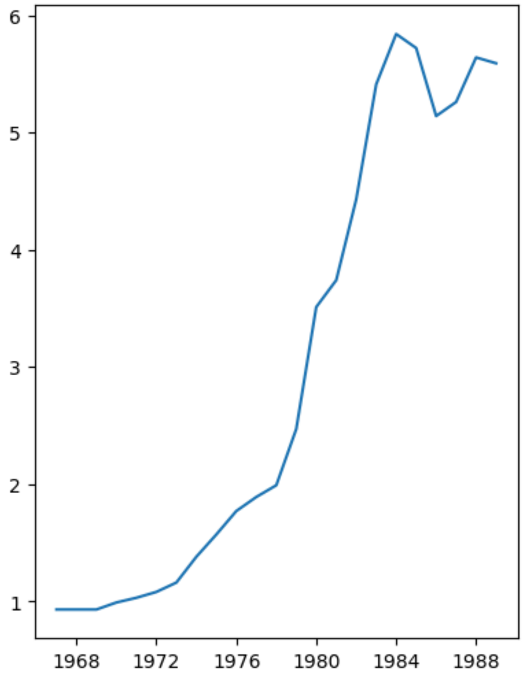
Out[11]:

	rownames	state	statecode	consumption	price	eprice	oprice	lprice	he
year									
1967-01-01	1	NY	35	313656	1.42	2.98	7.40	1.47	
1968-01-01	2	NY	35	319282	1.38	2.91	7.77	1.42	
1969-01-01	3	NY	35	331326	1.37	2.84	7.96	1.38	
1970-01-01	4	NY	35	346533	1.40	2.87	8.33	1.37	
1971-01-01	5	NY	35	352085	1.50	3.07	8.80	1.40	
...	...	...	...	...	...	...	...	...	
1985-01-01	134	CA	5	527495	5.72	7.78	30.58	5.84	
1986-01-01	135	CA	5	464307	5.14	7.95	44.15	5.72	
1987-01-01	136	CA	5	503473	5.26	8.03	35.24	5.14	
1988-01-01	137	CA	5	497138	5.64	8.69	34.02	5.26	
1989-01-01	138	CA	5	514276	5.59	9.45	44.44	5.64	

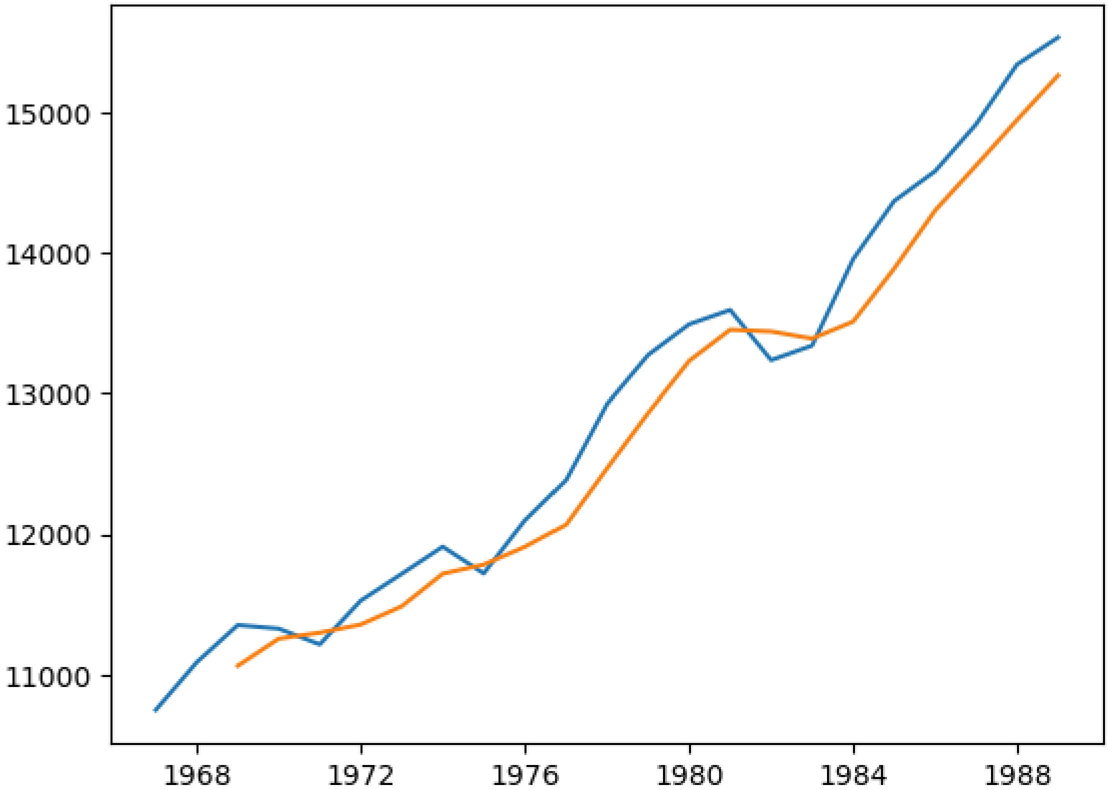
138 rows × 10 columns

In [12]:
```
# plotting time series of consumption and price for state CA
ca_data = ts_data[ts_data.state == "CA"]
plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
plt.plot(ca_data.consumption)
plt.subplot(1, 2, 2)
plt.plot(ca_data.price)
plt.show()
```

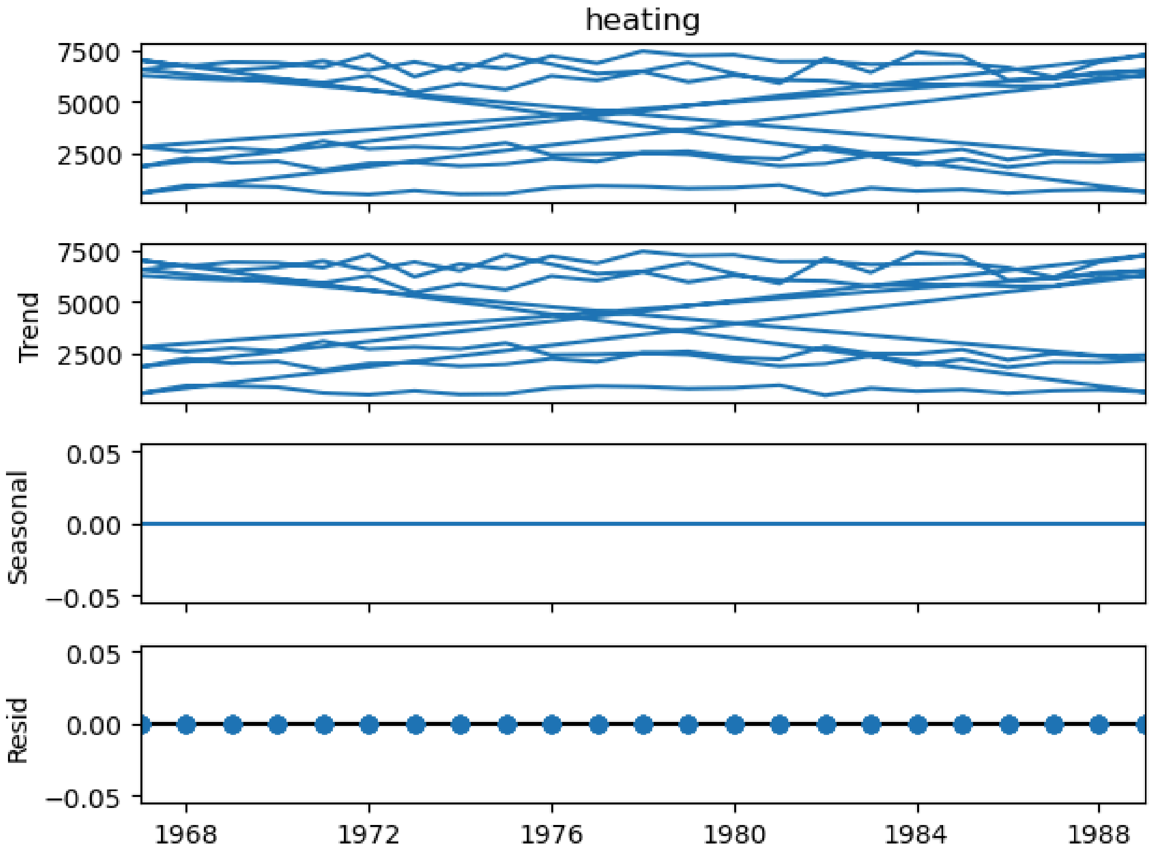




In [13]:
```
# moving average of income for state CA
ma = ca_data.income.rolling(window=3).mean()
plt.plot(ca_data.income)
plt.plot(ma)
plt.show()
```



In [14]:
```
# seasonal decomposition of heating for state CA
decomposed = seasonal_decompose(ts_data.heating, model='additive', period
decomposed.plot()
plt.show()
```



Module 4 - Data Summarisation and Visualisation

In [15]:
```
# 1D statistical analysis
# basic metrics
data.describe()
```

Out[15]:

	rownames	statecode	year	consumption	price	epri
count	138.000000	138.00000	138.000000	138.000000	138.000000	138.0000
mean	69.500000	27.00000	1978.000000	252901.478261	3.422319	5.0535
std	39.981246	15.68811	6.657415	184478.131559	2.169215	2.5778
min	1.000000	5.00000	1967.000000	9430.000000	0.680000	1.9800
25%	35.250000	10.00000	1972.000000	49103.500000	1.380000	2.4325
50%	69.500000	29.00000	1978.000000	300835.500000	2.775000	4.5200
75%	103.750000	44.00000	1984.000000	346428.750000	5.310000	7.2825
max	138.000000	45.00000	1989.000000	637289.000000	8.060000	10.8600

In [16]:
```
# kurtosis and skewness for consumption
kurt = data.consumption.kurt()
skew = data.consumption.skew()
print(f"{kurt}, {skew}")
```

```
-0.9746765850042207, 0.18689627507055256
```

In [17]:
```
# 2D statistical analysis
# correlation between consumption and income
```

```
c = data.consumption.corr(data.income)
print(f"{c}")
```

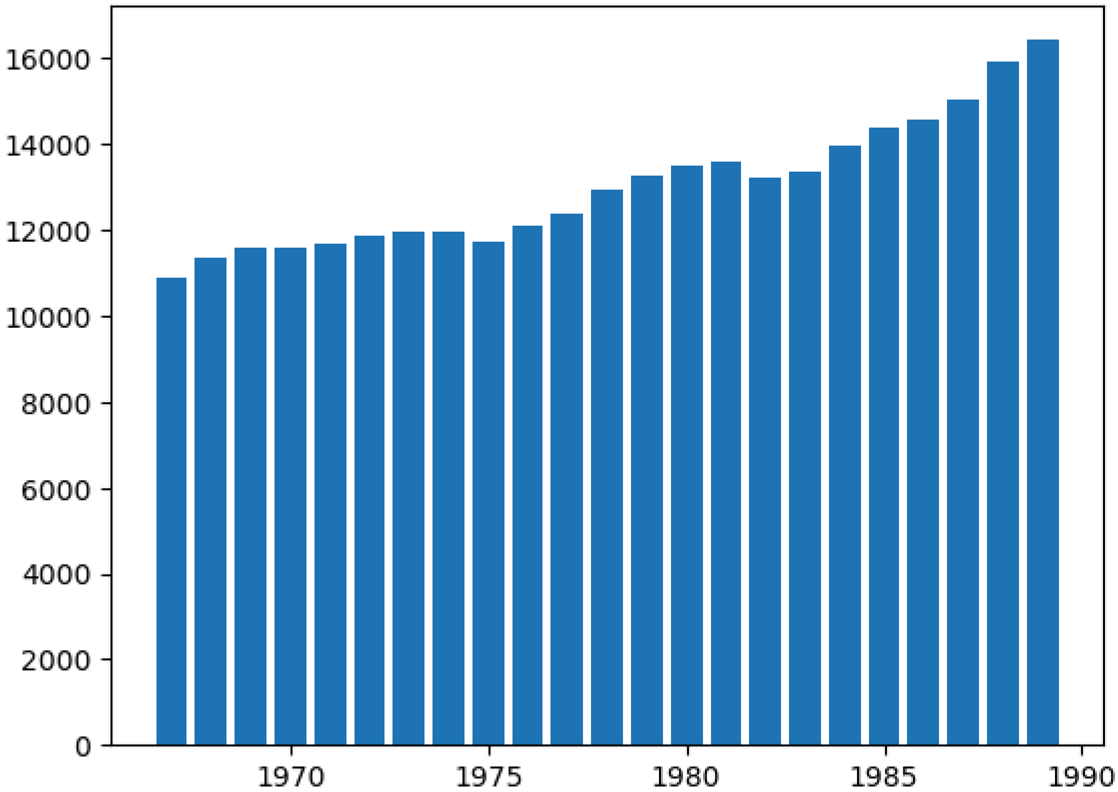
```
0.558558951233098
```

In [18]:
```
# covariance between price and income
c = data.price.cov(data.income)
print(f"{c}")
```

```
2723.491923294193
```

In [19]:
```
# bar plot between year and income
plt.bar(data.year, data.income)
```

Out[19]:
```
<BarContainer object of 138 artists>
```



Module 5 - Clustering Algorithms

In [20]:
```
# preprocessing data for clustering
data.drop("rownames", axis=1, inplace=True)

le = LabelEncoder()
state_encoded = le.fit_transform(data.state)
data.state = state_encoded
le.classes_
```

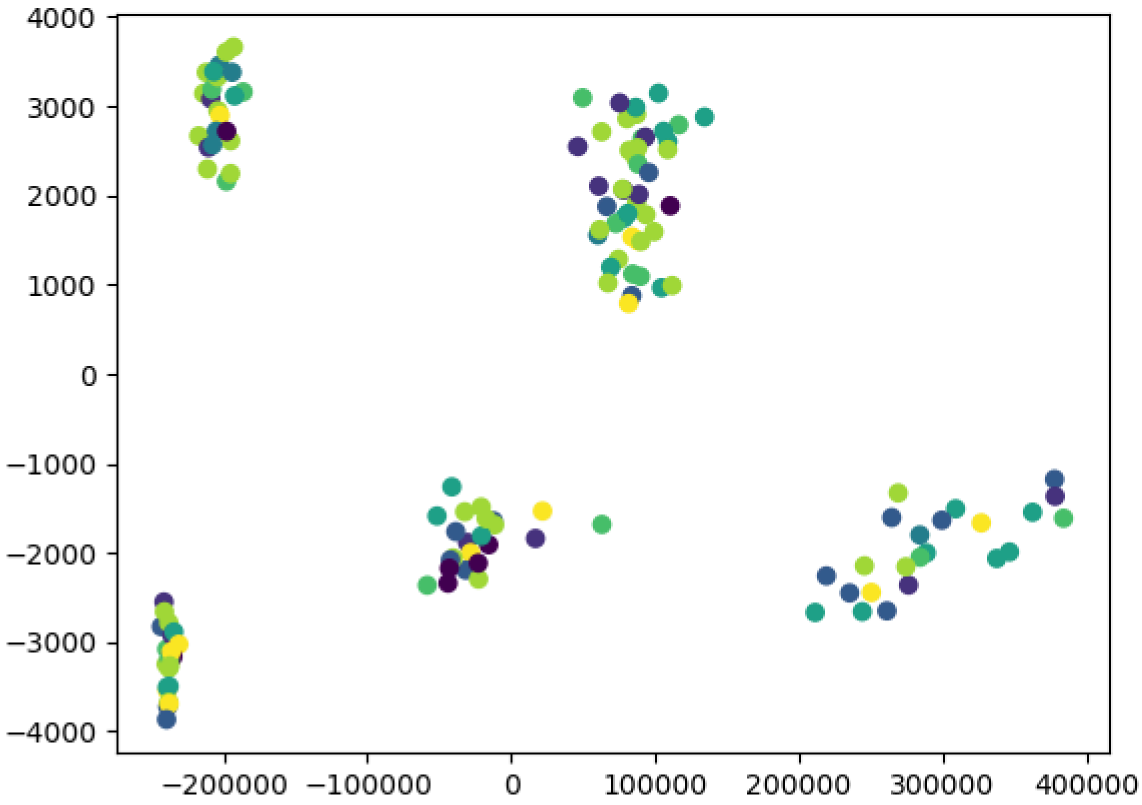
Out[20]:
```
array(['CA', 'FL', 'MI', 'NY', 'TX', 'UT'], dtype=object)
```

In [21]:
```
# reducing dimensions to plot
p = PCA(2)
reduced = p.fit_transform(data)
```

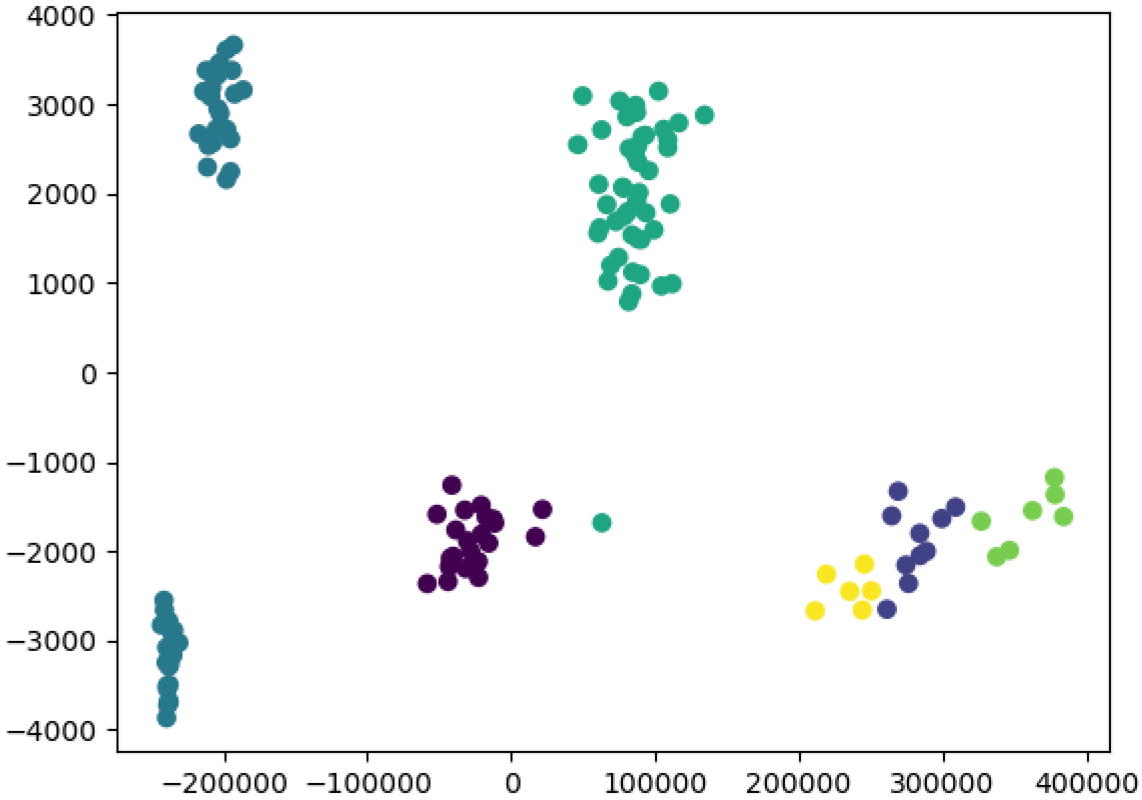
In [22]:
```
# spectral clustering
sc = SpectralClustering()
```

```
clusters = sc.fit_predict(data)

plt.scatter(reduced[:, 0], reduced[:, 1], c=clusters)
plt.show()
```

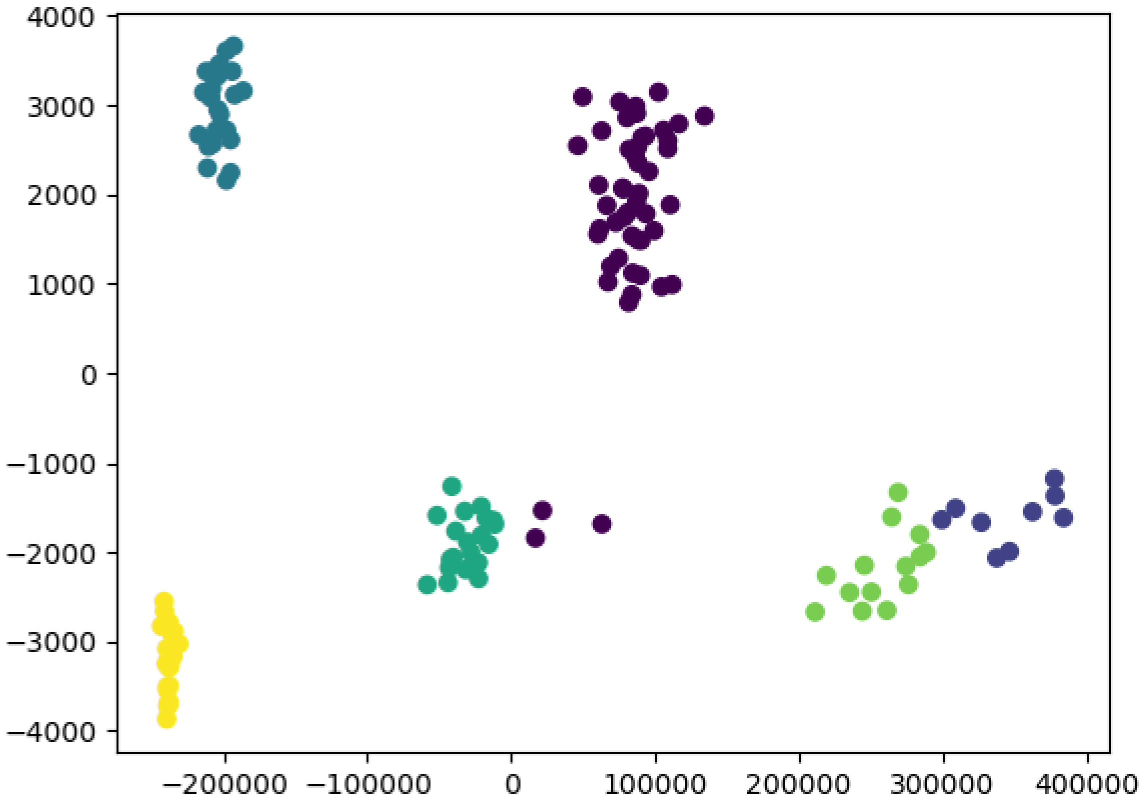
```
/Users/abhi/Programming/exploratory-data-analysis/env/lib/python3.12/site-
packages/sklearn/manifold/_spectral_embedding.py:329: UserWarning: Graph i
s not fully connected, spectral embedding may not work as expected.
  warnings.warn(
```



In [23]:
```
# k-means clustering
km = KMeans(6)
clusters = km.fit_predict(data)

plt.scatter(reduced[:, 0], reduced[:, 1], c=clusters)
plt.show()
```

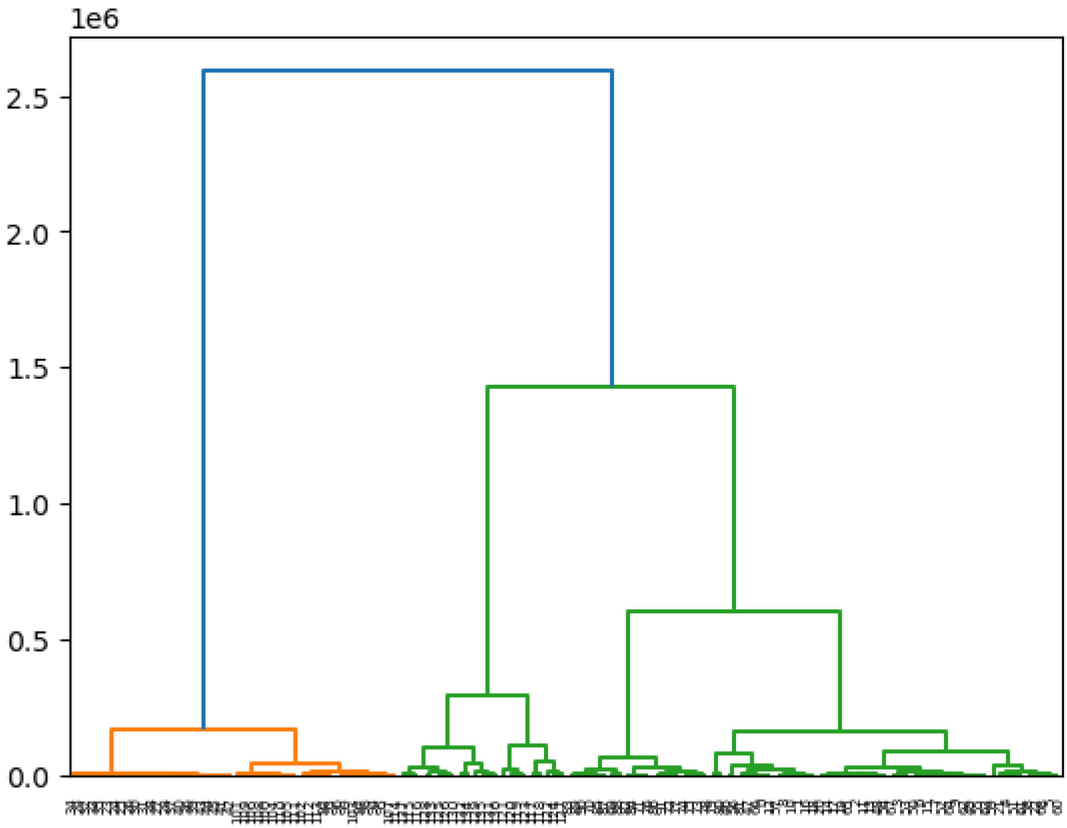


In [24]:
```
# agglomerative clustering
ac = AgglomerativeClustering(6, linkage="ward")
clusters = ac.fit_predict(data)

plt.scatter(reduced[:, 0], reduced[:, 1], c=clusters)
plt.show()
```



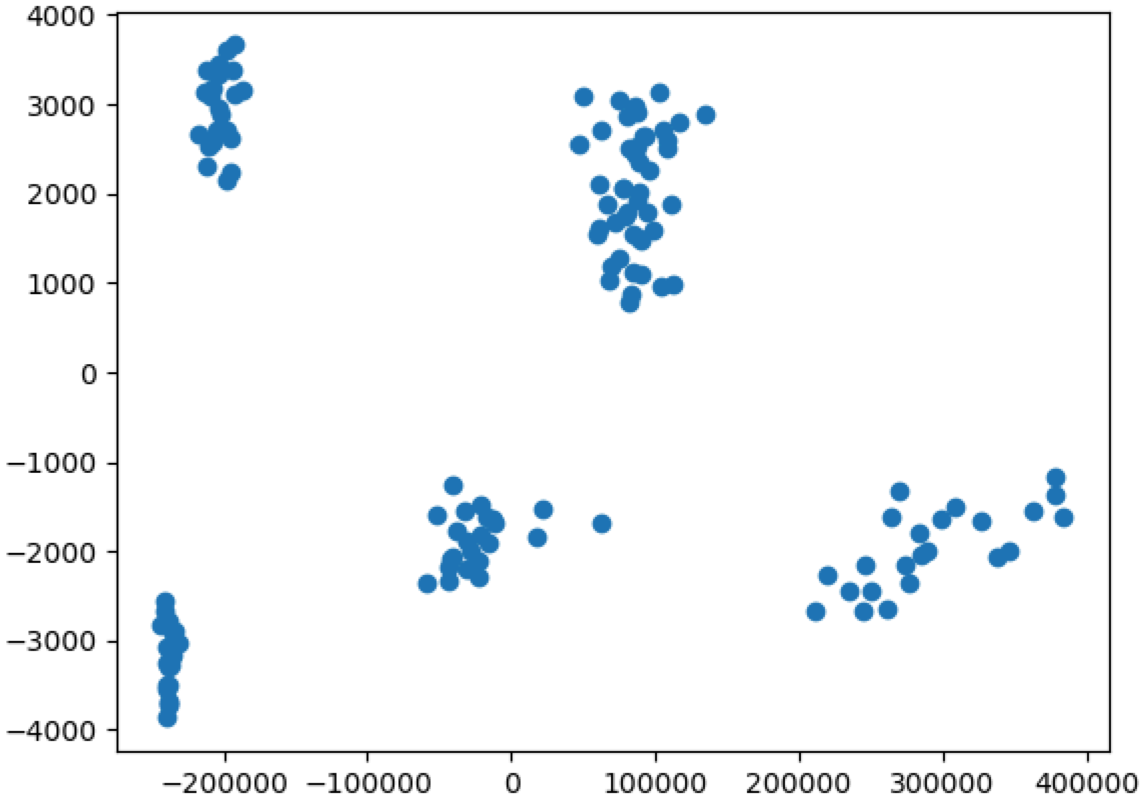
In [25]:
```
# dendrogram
links = linkage(data, method="ward")
```

```
dendrogram(links, show_leaf_counts=False)
plt.show()
```



Module 6 - Dimensionality Reduction

In [26]:
```
# principle component analysis - reducing to 2D
p = PCA(2)
reduced = p.fit_transform(data)

plt.scatter(reduced[:, 0], reduced[:, 1])
plt.show()
```



In [27]:
```
# singular value decomposition
U, S, VT = np.linalg.svd(data)
U.shape, S.shape, VT.shape
```

Out[27]: ((138, 138), (10,), (10, 10))

In [28]:
```
# reconstructing from singluar value decomposition
S_mat = np.zeros(data.shape)
np.fill_diagonal(S_mat, S)
reconstructed = np.dot(U, np.dot(S_mat, VT))
pd.DataFrame(reconstructed)
```

Out[28]:

	0	1	2	3	4	5	6	7	8	
0	3.000000e+00	35.0	1967.0	313656.0	1.42	2.98	7.40	1.47	6262.0	1090
1	3.000000e+00	35.0	1968.0	319282.0	1.38	2.91	7.77	1.42	6125.0	1137
2	3.000000e+00	35.0	1969.0	331326.0	1.37	2.84	7.96	1.38	6040.0	1157
3	3.000000e+00	35.0	1970.0	346533.0	1.40	2.87	8.33	1.37	6085.0	1158
4	3.000000e+00	35.0	1971.0	352085.0	1.50	3.07	8.80	1.40	5907.0	1165
...	...	...	...	...	...	...	...	...	...	
133	1.043889e-13	5.0	1985.0	527495.0	5.72	7.78	30.58	5.84	2694.0	1436
134	7.311560e-14	5.0	1986.0	464307.0	5.14	7.95	44.15	5.72	2192.0	1458
135	7.602678e-14	5.0	1987.0	503473.0	5.26	8.03	35.24	5.14	2502.0	1491
136	5.850017e-14	5.0	1988.0	497138.0	5.64	8.69	34.02	5.26	2366.0	1534
137	5.847746e-14	5.0	1989.0	514276.0	5.59	9.45	44.44	5.64	2420.0	1553

138 rows × 10 columns

In [29]:

```
# comparing it to original data
data
```

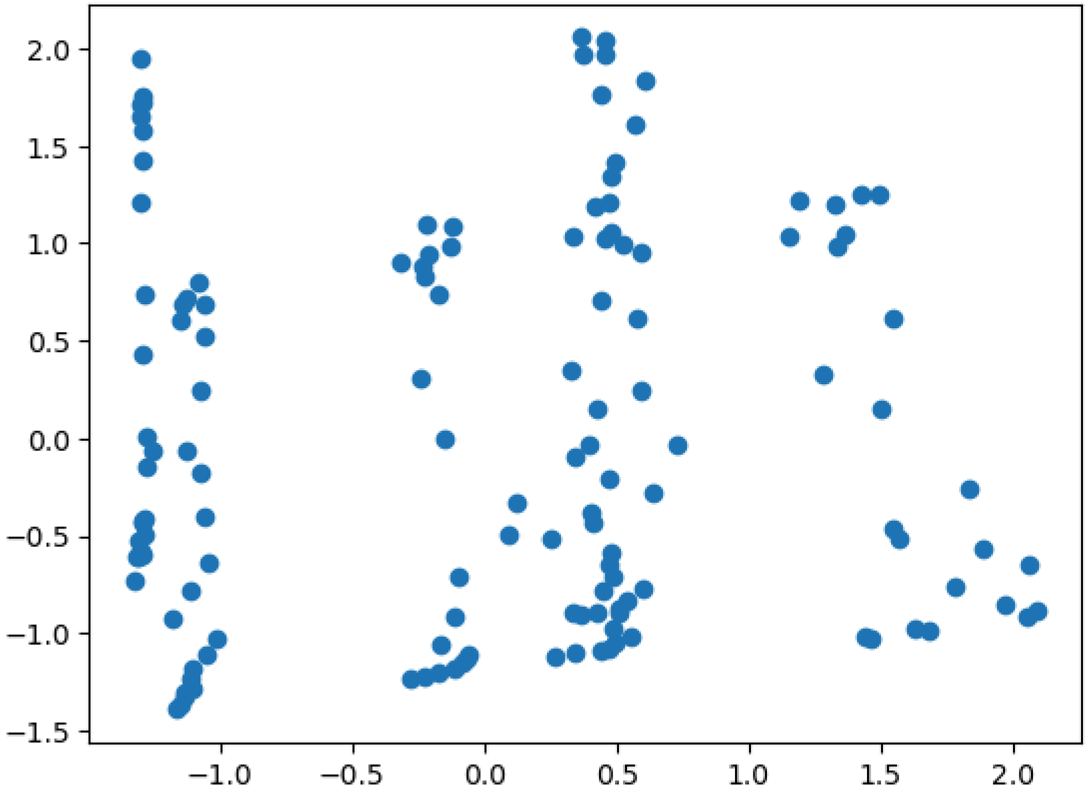
Out[29]:

	state	statecode	year	consumption	price	eprice	oprice	lprice	heating
0	3	35	1967	313656	1.42	2.98	7.40	1.47	6262
1	3	35	1968	319282	1.38	2.91	7.77	1.42	6125
2	3	35	1969	331326	1.37	2.84	7.96	1.38	6040
3	3	35	1970	346533	1.40	2.87	8.33	1.37	6085
4	3	35	1971	352085	1.50	3.07	8.80	1.40	5907
...	...	...	...	...	...	...	...	...	...
133	0	5	1985	527495	5.72	7.78	30.58	5.84	2694
134	0	5	1986	464307	5.14	7.95	44.15	5.72	2192
135	0	5	1987	503473	5.26	8.03	35.24	5.14	2502
136	0	5	1988	497138	5.64	8.69	34.02	5.26	2366
137	0	5	1989	514276	5.59	9.45	44.44	5.64	2420

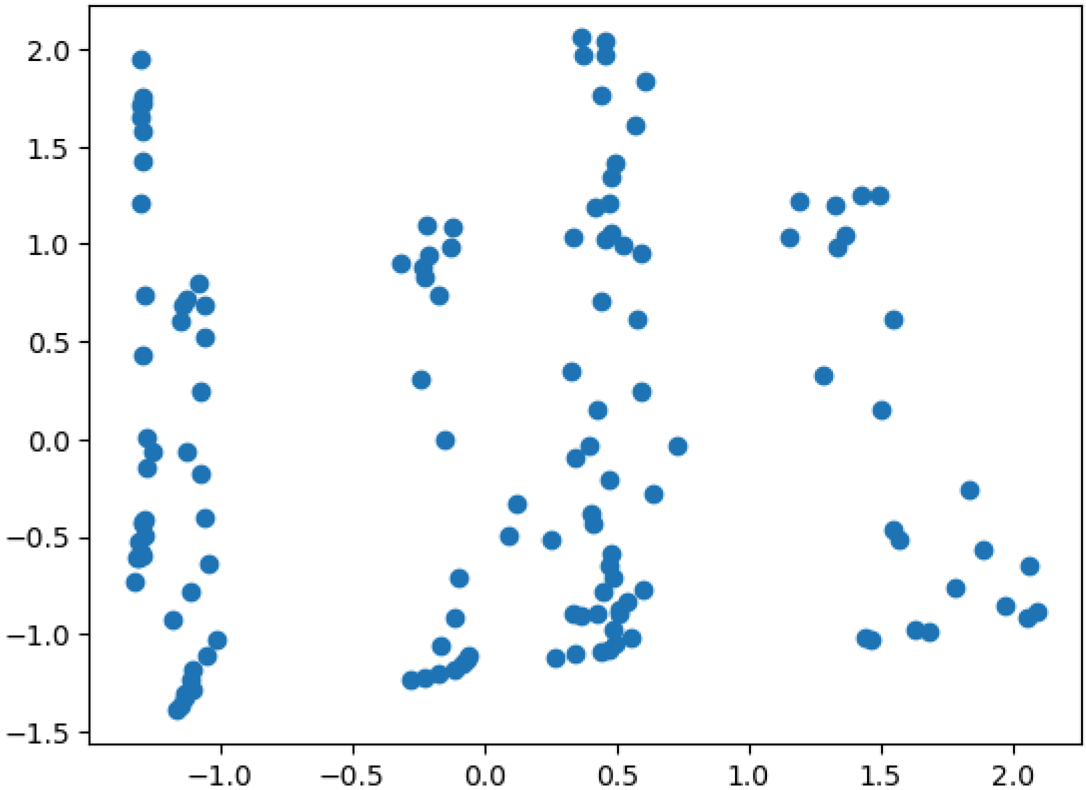
138 rows × 10 columns

In [30]:

```
# factor analysis
fa = FactorAnalysis(2)
factors = fa.fit_transform(data)

plt.scatter(factors[:, 0], factors[:, 1])
plt.show()
```

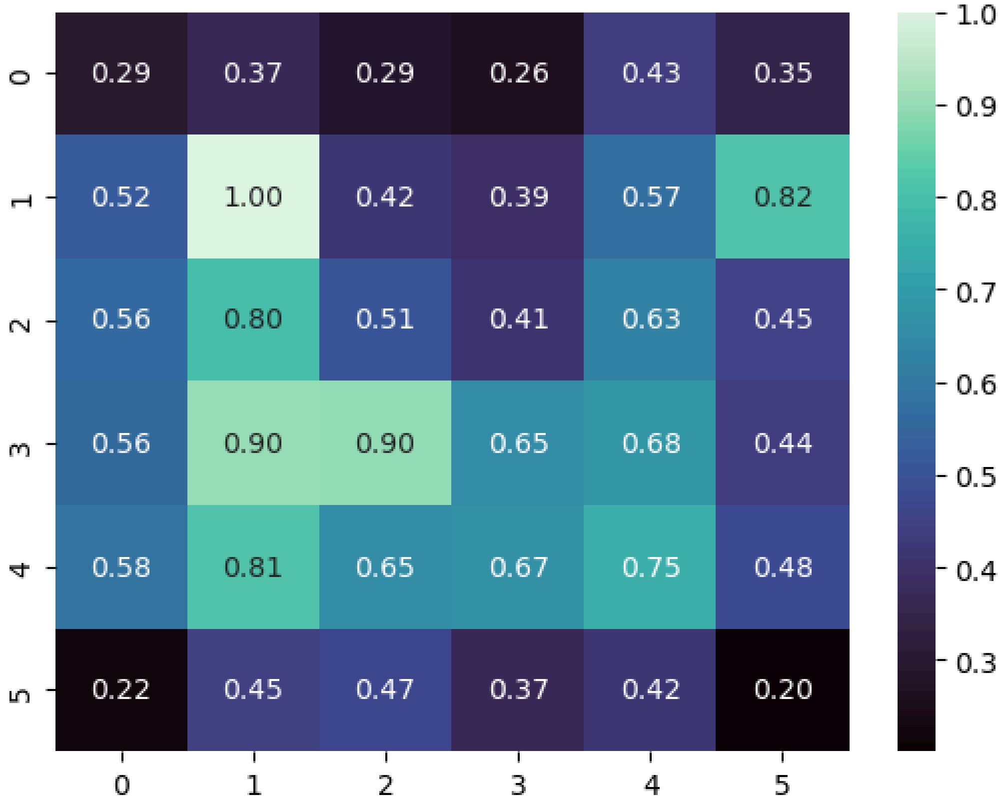


In [31]:
```
# multidimensional scaling
mds = MDS(2)
reduced = mds.fit_transform(data)

plt.scatter(factors[:, 0], factors[:, 1])
plt.show()
```



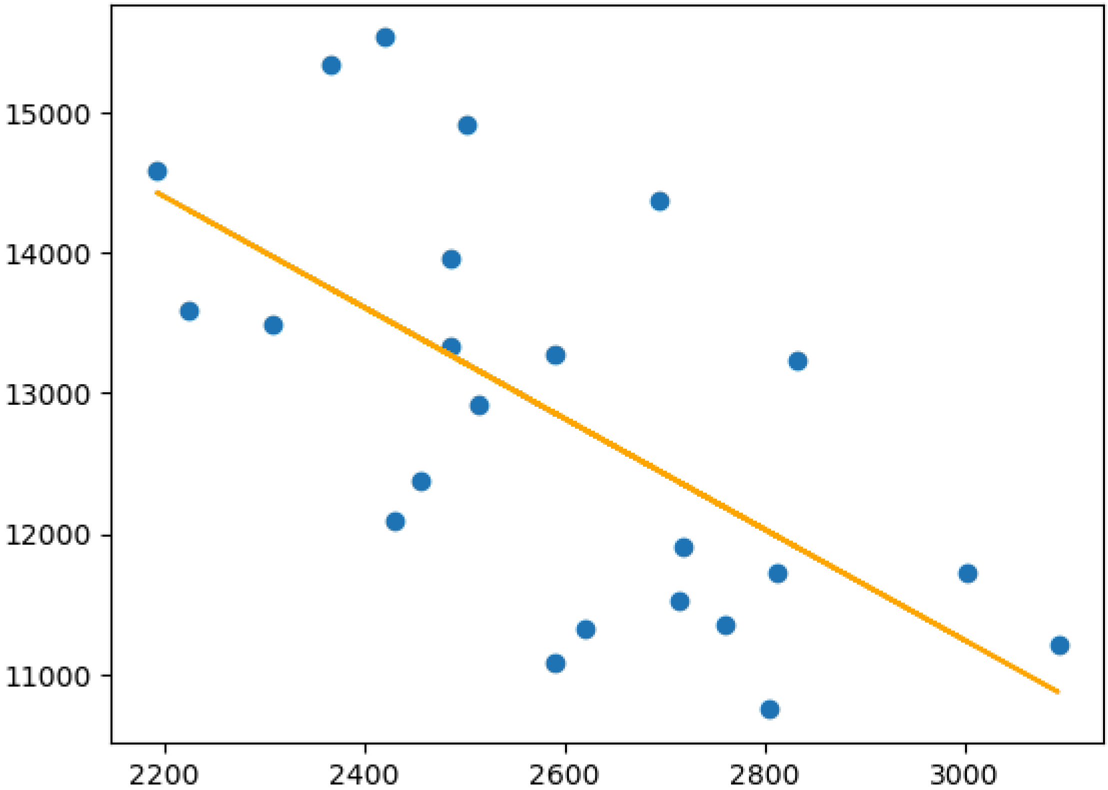
In [32]:
```
# scaling the data
sc = StandardScaler()
```

```
scaled = sc.fit_transform(data)

# self organising maps
size = 6
som = MiniSom(size, size, input_len=scaled.shape[1], sigma=1, learning_ra
som.random_weights_init(scaled)
som.train(scaled, 100)

U = som.distance_map().T
sns.heatmap(U, cmap="mako", annot=True, fmt=".2f")
plt.show()
```



Module 7 - Model Development and Evaluation

In [33]:
```
# linear regression between heating and income for state CA
lr = LinearRegression()
lr.fit(ca_data.heating.values.reshape(-1, 1), ca_data.income.values)
plt.scatter(ca_data.heating, ca_data.income)
plt.plot(ca_data.heating, lr.predict(ca_data.heating.values.reshape(-1, 1
plt.show()
```



In [34]:
```
# regression metrics
y_true = ca_data.income.values
y_pred = lr.predict(ca_data.heating.values.reshape(-1, 1))
mae = mean_absolute_error(y_true, y_pred)
rmse = root_mean_squared_error(y_true, y_pred)
r2 = r2_score(y_true, y_pred)

print(f"Mean absolute error: {mae}")
print(f"Root mean squared error: {rmse}")
print(f"R^2 score: {r2}")
```

```
Mean absolute error: 928.1284743763293
Root mean squared error: 1104.4346868510531
R^2 score: 0.39566198520333606
```