In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
import matplotlib.pyplot as plt
```

In [2]:

```
df=pd.read_csv('Global_Superstore2.csv', encoding = "ISO-8859-1")
df_unchanged=pd.read_csv('Global_Superstore2.csv', encoding = "ISO-8859-1")
```

Entfernen aller Attribute, die bei vorangegangener Exploration (wegen fehlender statistischer Abhängigkeit mit Profit) entfernt wurden

In [3]:

```
del df["State"],df_unchanged["State"]
```

In [4]:

```
del df["City"],df_unchanged["City"]
```

In [5]:

```
del df["Postal Code"],df_unchanged["Postal Code"]
```

In [6]:

```
del df["Product ID"],df_unchanged["Product ID"]
```

In [7]:

```
del df["Product Name"],df_unchanged["Product Name"]
```

In [8]:

```
del df["Order Priority"],df_unchanged["Order Priority"]
```

In [9]:

```
del df["Order ID"],df_unchanged["Order ID"]
```

In [10]:

```
del df["Ship Date"],df_unchanged["Ship Date"]
```

In [11]:

```
del df["Order Date"],df_unchanged["Order Date"]
```

In [12]:

```
del df["Ship Mode"],df_unchanged["Ship Mode"]
```

In [13]:

```
del df["Customer ID"],df_unchanged["Customer ID"]
```

In [14]:

```
del df["Customer Name"],df_unchanged["Customer Name"]
```

In [15]:

```
del df["Segment"],df_unchanged["Segment"]
```

Bereinigtes DataFrame:

In [16]:

```
df
```

Out[16]:

	Row ID	Country	Market	Region	Category	Sub-Category	Sales	Quantity	Discount
0	32298	United States	US	East	Technology	Accessories	2309.650	7	0.0
1	26341	Australia	APAC	Oceania	Furniture	Chairs	3709.395	9	0.1
2	25330	Australia	APAC	Oceania	Technology	Phones	5175.171	9	0.1
3	13524	Germany	EU	Central	Technology	Phones	2892.510	5	0.1
4	47221	Senegal	Africa	Africa	Technology	Copiers	2832.960	8	0.0
...	...	...	...	...	...	...	...	...	...
51285	29002	Japan	APAC	North Asia	Office Supplies	Fasteners	65.100	5	0.0
51286	35398	United States	US	Central	Office Supplies	Appliances	0.444	1	0.8
51287	40470	United States	US	West	Office Supplies	Envelopes	22.920	3	0.0
51288	9596	Brazil	LATAM	South	Office Supplies	Binders	13.440	2	0.0
51289	6147	Nicaragua	LATAM	Central	Office Supplies	Paper	61.380	3	0.0

51290 rows × 11 columns

Mit diesem DataFrame kann jetzt der Algorithmus/das Modell aufgesetzt werden, um den Profit vorherzusagen

1. Umwandeln der kategorischen Variablen in numerische

Diese Umwandlung ist nötig, da das Machine Learning Modell keine Zusammenhänge zwischen Text-Variablen finden kann. Durch die folgenden Methoden wird jeder kategorischen Ausprägung ein numerischer Wert zugeordnet.

In [17]:

```
df['Category'] =df['Category'].astype('category').cat.codes
df['Sub-Category'] =df['Sub-Category'].astype('category').cat.codes
df['Country'] =df['Country'].astype('category').cat.codes
df['Market'] =df['Market'].astype('category').cat.codes
df['Region'] =df['Region'].astype('category').cat.codes
```

In [18]:

```
df
```

Out[18]:

	Row ID	Country	Market	Region	Category	Sub-Category	Sales	Quantity	Discount	
0	32298	139	6	6	2	0	2309.650	7	0.0	76
1	26341	6	0	9	0	5	3709.395	9	0.1	-28
2	25330	6	0	9	2	13	5175.171	9	0.1	91
3	13524	47	4	3	2	13	2892.510	5	0.1	-9
4	47221	110	1	0	2	6	2832.960	8	0.0	31
...	...	...	...	...	...	...	...	...	...	
51285	29002	65	0	8	1	8	65.100	5	0.0	
51286	35398	139	6	3	1	1	0.444	1	0.8	-
51287	40470	139	6	12	1	7	22.920	3	0.0	1
51288	9596	17	5	10	1	3	13.440	2	0.0	
51289	6147	92	5	3	1	12	61.380	3	0.0	

51290 rows × 11 columns

Beim Vergleich mit dem ursprünglichen DataFrame fällt also auf, dass z.B. bei country dem Land Australien die Zahl 6 und dem Land Deutschland die Zahl 47 zugeordnet wurde.

2. Aufteilen des Datensatzes in Trainings- und Testdaten

In [19]:

```
X = np.asarray(df[['Country','Market','Region','Category','Sub-Category','Sales','Quantity'
Y = np.asarray(df['Profit'])

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, shuffle= True)
```

Die tran_test_split library von sklearn setzt das korrekte Splitting gemäß der Machine Learning Grundsätze bereits um (z.B. geeignetes Verhältnis zwischen Trainings- und Testdaten, Kreuzvalidierung (X-Val), ...)

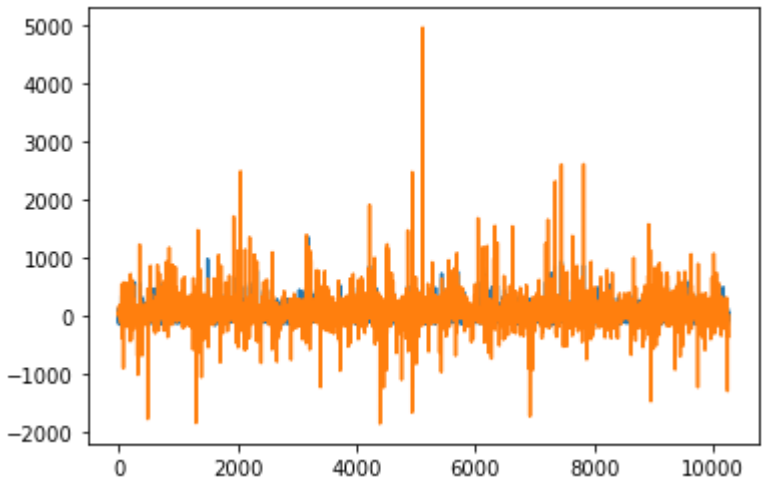
3. Lineare Regression

In [20]:

```
lineReg = LinearRegression()
lineReg.fit(X_train, y_train)
print('Score: ', lineReg.score(X_test, y_test))
print('Weights: ', lineReg.coef_)

plt.plot(lineReg.predict(X_test))
plt.plot(y_test,)
plt.show()
```

```
Score:  0.39683788718726243
Weights:  [ 8.91633850e-02 -6.99990820e-01 -3.06847892e-02  9.61649976e+00
 -1.72897480e+00  1.73699830e-01 -3.73935583e+00 -2.33029303e+02
 -1.06868240e-01]
```



Der Score spiegelt hier die Genauigkeit des Modells wider. Je höher der Score, desto besser das Modell.

Die Weights sind Gewichte, die den einzelnen Attributen zugeordnet werden, um den Profit vorherzusagen.

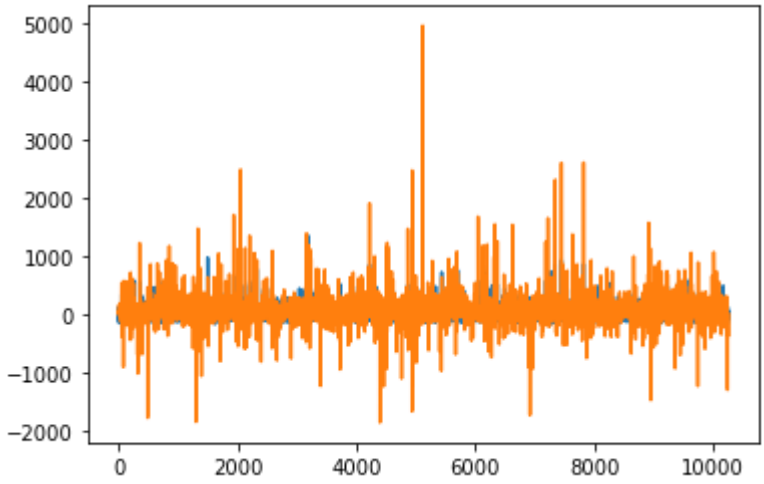
4. Ridge Regression

In [21]:

```
reg = linear_model.Ridge (alpha = .55)
reg.fit(X_train, y_train)
print('Score: ', reg.score(X_test, y_test))
print('Weights: ', reg.coef_)

plt.plot(reg.predict(X_test))
plt.plot(y_test)
plt.show()
```

```
Score:  0.3968412280035851
Weights:  [ 8.90796422e-02 -6.99088275e-01 -3.06819320e-02  9.61732795e+00
 -1.72898148e+00  1.73701709e-01 -3.73944843e+00 -2.32955389e+02
 -1.06859337e-01]
```



5. Vergleich: Welches Regressionsmodell passt besser?

Die folgenden Methoden durchlaufen das Modell für eine übergebene Anzahl von Iterationen. Grundsätzlich gilt, dass eine höhere Zahl Iterationen für eine höhere Aussagekraft des Ergebnisses spricht. Es wird also verglichen wie sich die Scores der beiden Modelle bei verschiedenen Anzahlen von Iterationen verhalten, um zu beurteilen welches Modell besser performt.

In [22]:

```
def ScoreLinearRegression(pAnzahl):
    scores = []
    coefs = []
    for i in range(pAnzahl):
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, shuffle= T
        lineReg = LinearRegression()
        lineReg.fit(X_train, y_train)
        scores.append(lineReg.score(X_test, y_test))
        coefs.append(lineReg.coef_)
    print('Linear Regression')
    print(np.mean(scores))
    print(np.mean(coefs, axis=0))
```

In [23]:

```
def ScoreRidgeRegression(pAnzahl):
    scores = []
    coefs = []
    for i in range(pAnzahl):
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, shuffle= T
        lineReg = linear_model.Ridge (alpha = .55)
        lineReg.fit(X_train, y_train)
        scores.append(lineReg.score(X_test, y_test))
        coefs.append(lineReg.coef_)
    print('\nRidge Regression')
    print(np.mean(scores))
    print(np.mean(coefs, axis=0))
```

Dass bei jedem Durchlauf der Datensatz neu "geshuffelt" wird und davon dann zufällige 20% als Testdaten genutzt werden, ist bei jeder Iteration die Aufteilung anders. Dies entspricht den Kriterien der Kreuzvalidierung und sorgt somit für valide Ergebnisse.

In [24]:

```
ScoreLinearRegression(50)
ScoreRidgeRegression(50)
```

```
Linear Regression
0.31240357058577256
[ 9.78666207e-02 -5.28961092e-01 -1.54944813e-01  1.00185531e+01
 -1.72164476e+00  1.82879208e-01 -3.58051013e+00 -2.30248098e+02
 -1.48116711e-01]

Ridge Regression
0.3090290776388641
[ 9.93115619e-02 -4.93073009e-01 -1.72400208e-01  1.01520050e+01
 -1.71811122e+00  1.83237059e-01 -3.58045728e+00 -2.29558956e+02
 -1.45249720e-01]
```

In [25]:

```
ScoreLinearRegression(100)
ScoreRidgeRegression(100)
```

```
Linear Regression
0.3035030564787257
[ 9.80056258e-02 -5.19789580e-01 -1.68587438e-01  1.00318531e+01
 -1.70785898e+00  1.84354288e-01 -3.58223977e+00 -2.29601564e+02
 -1.56827571e-01]

Ridge Regression
0.29379996368257594
[ 9.85039274e-02 -5.14611346e-01 -1.76271460e-01  1.01550898e+01
 -1.71626624e+00  1.85799484e-01 -3.62846819e+00 -2.29503033e+02
 -1.62994525e-01]
```

In [26]:

```
ScoreLinearRegression(500)
ScoreRidgeRegression(500)
```

```
Linear Regression
0.30912789583584777
[ 9.82157910e-02 -5.08476697e-01 -1.72909838e-01  1.00704897e+01
 -1.71409865e+00  1.83081622e-01 -3.59893570e+00 -2.29988501e+02
 -1.45872929e-01]

Ridge Regression
0.2984647228896093
[ 9.78220033e-02 -4.82541627e-01 -1.67286975e-01  1.00323946e+01
 -1.71726027e+00  1.85944987e-01 -3.67603170e+00 -2.29747919e+02
 -1.61869812e-01]
```

In [27]:

```
ScoreLinearRegression(1000)
ScoreRidgeRegression(1000)
```

```
Linear Regression
0.3058513107363537
[ 9.80113160e-02 -5.06903860e-01 -1.68440588e-01  1.00794544e+01
 -1.71127763e+00  1.84246435e-01 -3.62273833e+00 -2.29942456e+02
 -1.53041553e-01]

Ridge Regression
0.3088750256629237
[ 9.79686545e-02 -4.99803692e-01 -1.69193267e-01  1.00698312e+01
 -1.70912195e+00  1.83351837e-01 -3.59823313e+00 -2.29894312e+02
 -1.48215470e-01]
```

In [28]:

```
ScoreLinearRegression(2000)
ScoreRidgeRegression(2000)
```

```
Linear Regression
0.3067850526031851
[ 9.82775277e-02 -5.14586267e-01 -1.66467117e-01  1.00791538e+01
 -1.70946934e+00  1.83673947e-01 -3.60545447e+00 -2.30010644e+02
 -1.49531612e-01]

Ridge Regression
0.3062929273385414
[ 9.80171925e-02 -5.11887046e-01 -1.66689928e-01  1.00669521e+01
 -1.70879139e+00  1.83518580e-01 -3.61048446e+00 -2.29788146e+02
 -1.47325952e-01]
```

In [50]:

```
ScoreLinearRegression(10000)
ScoreRidgeRegression(10000)
```

```
Linear Regression
0.30456519263555165
[ 9.81977235e-02 -5.07382033e-01 -1.70108314e-01  1.00781184e+01
 -1.71110504e+00  1.84274744e-01 -3.61490693e+00 -2.29847549e+02
 -1.52599483e-01]

Ridge Regression
0.30677536888281814
[ 9.80418739e-02 -5.11366525e-01 -1.68522859e-01  1.00681385e+01
 -1.71085939e+00  1.83704621e-01 -3.60505142e+00 -2.29862475e+02
 -1.49474436e-01]
```

Beide Modelle sind sehr ähnlich, aufgrund ihres gleichen Ansatzes. Dennoch zeigt sich bei einer großen Anzahl von Iterationen, dass die Ridge Regression etwas besser ist. Daher nutzen wir im Folgenden die Ridge Regression als finales Modell.

6. Tabellen aufstellen, um gewünschte Inputs übergeben zu können

In [29]:

```
def tabelleErzeugen(pColumn):
    g = df_unchanged.groupby(pColumn)
    array1=list(g.groups.keys())
    g = df.groupby(pColumn)
    array2=list(g.groups.keys())
    Kategorien = pd.DataFrame(list(zip(array1, array2)),columns =['Name', 'Zahl'])
    return Kategorien
```

In [30]:

```
def getCountryNumber(pCountry):
    a=tabelleErzeugen("Country")
    return a.loc[a['Name'] == pCountry]
```

Diese Methode gibt das DataFrame zurück, in dem zu sehen ist, welche Zahl für welche Ausprägung der jeweiligen kategorischen Variable steht.

In [31]:

```
tabelleErzeugen("Country")
```

Out[31]:

	Name	Zahl
0	Afghanistan	0
1	Albania	1
2	Algeria	2
3	Angola	3
4	Argentina	4
...	...	...
142	Venezuela	142
143	Vietnam	143
144	Yemen	144
145	Zambia	145
146	Zimbabwe	146

147 rows × 2 columns

Hier sind nicht alle Werte sichtbar. Durch den Folgenden Aufruf kann die Zahl spezifischer Länder abgefragt werden: getCountryNumber(pLand) Beispiel:

In [32]:

```
getCountryNumber("Germany")
```

Out[32]:

	Name	Zahl
47	Germany	47

In [33]:

```
tabelleErzeugen("Market")
```

Out[33]:

	Name	Zahl
0	APAC	0
1	Africa	1
2	Canada	2
3	EMEA	3
4	EU	4
5	LATAM	5
6	US	6

In [34]:

```
tabelleErzeugen("Region")
```

Out[34]:

	Name	Zahl
0	Africa	0
1	Canada	1
2	Caribbean	2
3	Central	3
4	Central Asia	4
5	EMEA	5
6	East	6
7	North	7
8	North Asia	8
9	Oceania	9
10	South	10
11	Southeast Asia	11
12	West	12

In [35]:

```
tabelleErzeugen("Category")
```

Out[35]:

	Name	Zahl
0	Furniture	0
1	Office Supplies	1
2	Technology	2

In [36]:

```
tabelleErzeugen("Sub-Category")
```

Out[36]:

	Name	Zahl
0	Accessories	0
1	Appliances	1
2	Art	2
3	Binders	3
4	Bookcases	4
5	Chairs	5
6	Copiers	6
7	Envelopes	7
8	Fasteners	8
9	Furnishings	9
10	Labels	10
11	Machines	11
12	Paper	12
13	Phones	13
14	Storage	14
15	Supplies	15
16	Tables	16

7. Mit dem Modell den Profit vorhersagen

Jetzt kann das Modell dazu genutzt werden anhand verschiedener Parameter den Profit der anstehenden Transaktion vorherzusagen.

Es muss einfach der Befehl **reg.predict([['Country','Market','Region','Category','Sub-Category','Sales','Quantity','Discount','Shipping Cost']])** ausgeführt werden. Die kategorischen Variablen

werden durch die im 6. Schritt ermittelten Ersatzzahlen eingegeben und für die numerischen Werte werden einfach die tatsächlichen Zahlen übergeben.

Es folgt ein Anwendungsbeispiel

7.1 Ein Unternehmen möchte ein Handy verkaufen. Es soll nur einzeln und nicht in Mengenangebot verpackt werden. Es soll maximal Rabatte von 10 % geben. Jetzt ist die Frage was der beste Absatzmarkt ist. Unser Modell wird genutzt, um zu prüfen für welche geografischen Parameter der Profit am größten ist. Die Shipping Costs sind abhängig vom Markt, weil das Unternehmen seinen Sitz in den USA hat. Es werden also zwischen den Varianten die Attribute Country, Region, Market und Shipping Cost verändert.

1. Variante:

Country: 139 (United States), Market: 6 (US), Region: 7 (North), Category: 2 (Technology), Sub-Category: 13 (Phones), Quantity: 1, Discount: 0.1, Shipping Cost: 400

In [37]:

```
getCountryNumber("United States")
```

Out[37]:

	Name	Zahl
139	United States	139

In [38]:

```
reg.predict([[139, 6, 7, 2, 13, 1000, 1, 0.1, 400]])
```

Out[38]:

```
array([141.97011289])
```

2. Variante:

Country: 139 (United States), Market: 6 (US), Region: 6 (East), Category: 2 (Technology), Sub-Category: 13 (Phones), Quantity: 1, Discount: 0.1, Shipping Cost: 400

In [39]:

```
reg.predict([[139, 6, 6, 2, 13, 1000, 1, 0.1, 400]])
```

Out[39]:

```
array([142.00079483])
```

3. Variante:

Country: 139 (United States), Market: 6 (US), Region: 12 (West), Category: 2 (Technology), Sub-Category: 13 (Phones), Quantity: 1, Discount: 0.1, Shipping Cost: 400

In [40]:

```
reg.predict([[139, 6, 12, 2, 13, 1000, 1, 0.1, 400]])
```

Out[40]:

```
array([141.81670323])
```

Die Region innerhalb der USA macht also keinen Unterschied. Daher werden jetzt Absatzmärkte auf einem anderen Kontinent ausprobiert

4. Variante:

Country: 17 (Brazil), Market: 5 (LATAM), Region: 10 (South), Category: 2 (Technology), Sub-Category: 13 (Phones), Quantity: 1, Discount: 0.1, Shipping Cost: 700

In [41]:

```
getCountryNumber("Brazil")
```

Out[41]:

	Name	Zahl
17	Brazil	17

In [42]:

```
reg.predict([[17, 5, 10, 2, 13, 1000, 1, 0.1, 700]])
```

Out[42]:

```
array([99.65163798])
```

In [43]:

```
df_unchanged.loc[df_unchanged['Region'] == "Central Asia"]
```

Out[43]:

	Row ID	Country	Market	Region	Category	Sub-Category	Sales	Quantity	Discount
11	28879	Afghanistan	APAC	Central Asia	Furniture	Tables	4626.15	5	0.0
29	22999	India	APAC	Central Asia	Furniture	Chairs	1878.72	4	0.0
41	29272	India	APAC	Central Asia	Technology	Phones	4518.78	7	0.0
42	25795	India	APAC	Central Asia	Furniture	Bookcases	5667.87	13	0.0
48	28701	India	APAC	Central Asia	Technology	Machines	2174.13	7	0.0
...	...	...	...	...	...	...	...	...	...
51103	20407	India	APAC	Central Asia	Office Supplies	Labels	6.27	1	0.0
51163	25350	India	APAC	Central Asia	Office Supplies	Art	58.44	2	0.0
51235	24004	Pakistan	APAC	Central Asia	Office Supplies	Labels	17.28	4	0.5
51253	29192	Pakistan	APAC	Central Asia	Office Supplies	Paper	18.36	2	0.5
51283	24105	India	APAC	Central Asia	Office Supplies	Paper	26.94	2	0.0

2048 rows × 11 columns

5. Variante:

Country: 47 (Germany), Market: 4 (EU), Region: 3 (Central), Category: 2 (Technology), Sub-Category: 13 (Phones), Quantity: 1, Discount: 0.1, Shipping Cost: 650

In [44]:

```
reg.predict([[47, 4, 3, 2, 13, 1000, 1, 0.1, 650]])
```

Out[44]:

```
array([108.58085588])
```

6. Variante:

Country: 110 (Senegal), Market: 1 (Africa), Region: 0 (Africa), Category: 2 (Technology), Sub-Category: 13 (Phones), Quantity: 1, Discount: 0.1, Shipping Cost: 1200

In [45]:

```
getCountryNumber("Senegal")
```

Out[45]:

	Name	Zahl
110	Senegal	110

In [46]:

```
reg.predict([[110, 1, 0, 2, 13, 1000, 1, 0.1, 1200]])
```

Out[46]:

```
array([57.60954871])
```

7. Variante:

Country: 26 (China), Market: 0 (APAC), Region: 8 (North Asia), Category: 2 (Technology), Sub-Category: 13 (Phones), Quantity: 1, Discount: 0.1, Shipping Cost: 950

In [47]:

```
getCountryNumber("China")
```

Out[47]:

	Name	Zahl
26	China	26

In [48]:

```
reg.predict([[26, 0, 8, 2, 13, 1000, 1, 0.1, 950]])
```

Out[48]:

```
array([77.29532579])
```

Fazit des Beispiels

Der beste Absatzmarkt für das Handy des Unternehmens scheint die USA selbst zu sein. Aufgrund der erhöhten Shipping Costs bei Verkauf in anderen Kontinenten, ist **der vorhergesagte Profit in den USA mit Abtsand am höchsten.** Welcher konkrete Markt innerhalb der USA genutzt wird, spielt keine Rolle. Das Handy würde in allen gleich gut laufen.

In [ ]:

```

```