Data Cleaning

Features

source_id = producer code —> only got 1 —> useless
hs_code = list of numbers used by customs to classify a product
commodity_desc = product name
geography_code = location code
geography_desc = location name (paired with location code)
attribute_desc = export quantity or export value
unit_desc = KG, $ or L
year_id = year
timeperiod_id = month
amount = amount transacted

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

In [2]:
```
df_product = pd.read_csv('catfish_trout.csv')
```

In [3]:
```
def catfish_filter(val):
    catfish_stat = re.search(r'C*c*at',val)
    if catfish_stat:
        return True
    else:
        return False

#df_filtered = df[df['col'].apply(regex_filter)]
```

In [4]:
```
df_filtered = df_product[df_product['commodity_desc'].apply(catfish_filter)]
```

In [5]:
```
df_filtered = df_filtered[df_filtered['geography_desc'] == 'United States of America']
```

In [6]:
```
df_filtered.drop(columns = ['source_id', 'geography_desc', 'geography_code'], inplace=True)
df_filtered = df_filtered[df_filtered['attribute_desc'].isin(['Farm Sales to Processors', 'Farm Price', 'Producer
```

In [7]:
```
df_sales = df_filtered[df_filtered['attribute_desc'].isin(['Farm Sales to Processors'])]
df_price = df_filtered[df_filtered['attribute_desc'].isin(['Farm Price'])]
df_inventory = df_filtered[df_filtered['attribute_desc'].isin(['Producer Inventories'])]
```

In [8]:
```
df_price.rename(columns={'amount': 'price'}, inplace = True)
```

```
/Users/salimwid/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4441: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy
  return super().rename(
```

In [9]:
```
df_price.drop(columns=['attribute_desc','unit_desc'], inplace= True)
```

```
/Users/salimwid/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy
  return super().drop(
```

In [10]:
```
df_sales.drop(columns=['attribute_desc', 'unit_desc'], inplace=True)
```

In [11]:
```
df_sales_his = df_sales
df_sales_his = df_sales_his.merge(df_price, on = ['year_id', 'timeperiod_id', 'commodity_desc'])
df_sales = df_sales.merge(df_price, on = ['year_id', 'timeperiod_id', 'commodity_desc'])
df_sales = df_sales[df_sales['year_id'] != 2013]
```

In [12]:
```
year_value_list = pd.pivot_table(df_sales, values=['amount', 'price'], index=['year_id'], aggfunc={'amount': [np.s
```

In [13]:
```
year_dict = {'amount_agg':[], 'amount_mean': [],'price_mean': [], 'price_std': []}
```

In [14]:
```
for i in year_value_list:
    year_dict['amount_mean'].append(i[0])
    year_dict['amount_agg'].append(i[1])
    year_dict['price_mean'].append(i[2])
    year_dict['price_std'].append(i[3])
```

In [15]:
```
df_plot = pd.DataFrame.from_dict(year_dict)
```

In [16]:
```
df_plot['year'] = df_sales['year_id'].unique()
df_plot.set_index('year', inplace=True)
```

In [17]:
```
df_plot1 = df_plot[['price_mean', 'price_std']]
df_plot2 = df_plot[['amount_agg', 'amount_mean']]
```

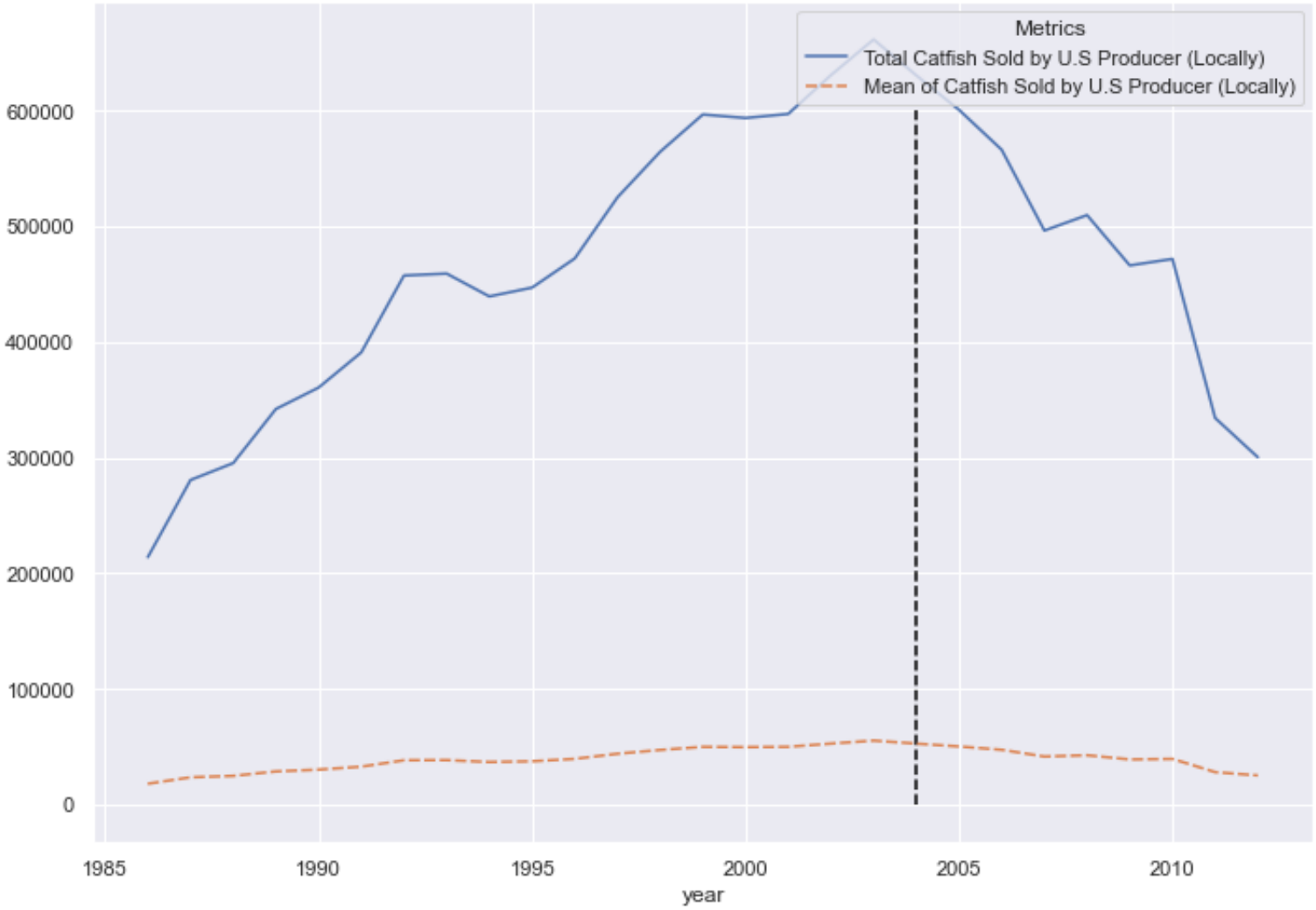
In [18]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.lineplot(data = df_plot1, legend=False)
plt.vlines(x=2004, color='black', linestyle='--', ymin = 0, ymax = 125)
plt.legend(title='Metrics', loc='upper right', labels=['Catfish Price Mean', 'Catfish Price Standard Deviation'])
```

Out[18]: <matplotlib.legend.Legend at 0x7fef97f264f0>

In [19]:

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.lineplot(data = df_plot2, legend=False)
plt.vlines(x=2004, color='black', linestyle='--', ymin = 0, ymax = 600000)
plt.legend(title='Metrics', loc='upper right', labels=['Total Catfish Sold by U.S Producer (Locally)', 'Mean of Ca
```

Out[19]: <matplotlib.legend.Legend at 0x7fef98a75f40>



In [20]:
```
pd.pivot_table(df_sales, values=['amount', 'price'], index=['year_id'], aggfunc={'amount': [np.sum,np.mean], 'pric
```

Out[20]:

	amount	price

	mean	sum	mean	std
year_id				
1986	17813.000000	213756.0	66.833333	4.745013
1987	23374.666667	280496.0	61.833333	3.214550
1988	24592.416667	295109.0	76.416667	3.604501
1989	28491.666667	341900.0	71.666667	5.613836
1990	30036.250000	360435.0	75.750000	2.179449
1991	32572.500000	390870.0	63.083333	5.583390
1992	38113.916667	457367.0	59.833333	3.069893
1993	38251.083333	459013.0	71.000000	3.074824
1994	36605.750000	439269.0	78.416667	1.975225
1995	37240.500000	446886.0	78.583333	0.514929
1996	39343.583333	472123.0	77.250000	1.815339
1997	43745.750000	524949.0	71.166667	1.850471
1998	47029.583333	564355.0	74.166667	3.737606
1999	49719.000000	596628.0	73.675000	2.637190
2000	49466.916667	593603.0	74.975000	3.918749
2001	49759.000000	597108.0	64.516667	5.313761
2002	52550.083333	630601.0	56.766667	1.479148
2003	55125.333333	661504.0	58.116667	3.410634
2004	52537.500000	630450.0	69.608333	1.866186
2005	50055.833333	600670.0	72.516667	0.348590

2006	47177.583333	566131.0	79.625000	4.146658
2007	41353.833333	496246.0	76.658333	7.723689
2008	42466.416667	509597.0	77.975000	5.789823
2009	38841.666667	466100.0	77.075000	1.296236
2010	39306.916667	471683.0	80.233333	2.990085
2011	27845.250000	334143.0	117.700000	11.669774
2012	25012.583333	300151.0	97.550000	18.793785

In [21]:
```
df_sales.set_index(['year_id', 'timeperiod_id'], inplace=True)
df_sales.reset_index(inplace=True)
```

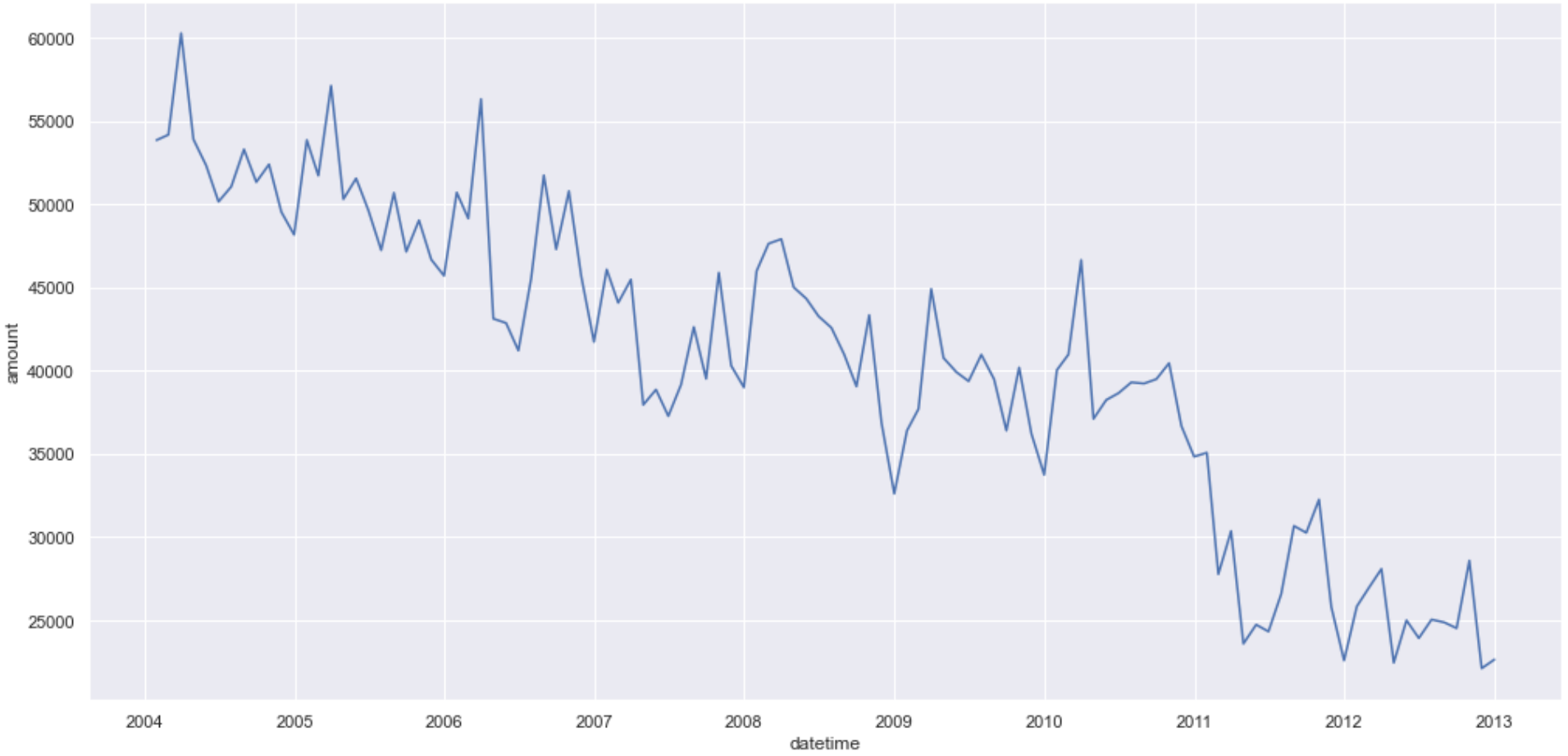
In [22]:
```
df_sales = df_sales[df_sales['year_id'] >= 2004]
```

In [23]:
```
df_sales['datetime'] = pd.date_range(start='1/1/2004', periods=108, freq='M')
```

In [24]:
```
df_sales.reset_index(inplace=True)
df_sales.set_index('datetime', inplace=True)
```

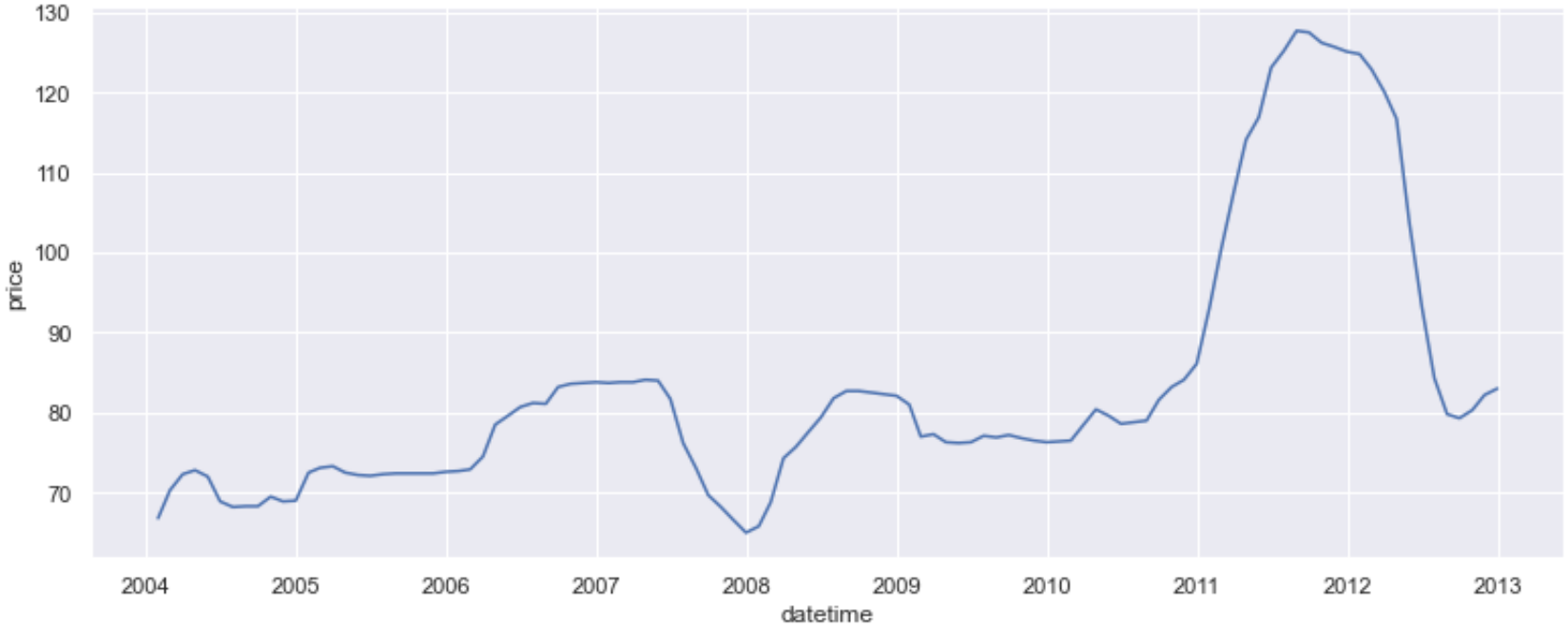
In [25]:
```
sns.set(rc={'figure.figsize':(17,8.27)})
sns.lineplot(data=df_sales, x='datetime', y='amount')
```

Out[25]: <AxesSubplot:xlabel='datetime', ylabel='amount'>



In [26]:
```
sns.set(rc={'figure.figsize':(13,5)})
sns.lineplot(data=df_sales, x='datetime', y='price')
```

Out[26]: <AxesSubplot:xlabel='datetime', ylabel='price'>



In [27]:
```
min_val = df_sales[['year_id', 'price']].groupby('year_id').min().values
max_val = df_sales[['year_id', 'price']].groupby('year_id').max().values
```

In [28]:
```
def calculate_benefit_val(df, min_max, colname, year, col_selected):
    for i in min_max:
        year+= 1
        df_sales.loc[df_sales['year_id'] == year, colname] = df_sales[col_selected] - i[0]
    return None

calculate_benefit_val(df_sales, min_val, 'benefit_criterion', 2003, 'price')
```

In [29]:
```
def calculate_criterion_val(df, min_max, colname, year, col_selected):
    for i in min_max:
        year+= 1
        df_sales.loc[df_sales['year_id'] == year, colname] = i[0] - df_sales[col_selected]
    return None

calculate_criterion_val(df_sales, max_val, 'regret_criterion', 2003, 'price')
```

In [30]:
```
df_sales['benefit_criterion'].values.reshape(9,12)
```

Out[30]:
```
array([[ 0. ,  3.5,  5.5,  6. ,  5.2,  2.1,  1.4,  1.5,  1.5,  2.7,  2.1,
         2.2],
       [ 0.4,  1. ,  1.2,  0.4,  0.1,  0. ,  0.2,  0.3,  0.3,  0.3,  0.3,
         0.5],
       [ 0. ,  0.2,  1.8,  5.8,  6.9,  8. ,  8.5,  8.4, 10.5, 10.9, 11. ,
        11.1],
       [18.7, 18.8, 18.8, 19.1, 19. , 16.7, 11.2,  8.1,  4.7,  3.2,  1.6,
         0. ],
       [ 0. ,  3. ,  8.5,  9.9, 11.8, 13.6, 16. , 16.9, 16.9, 16.7, 16.5,
        16.3],
       [ 4.8,  0.8,  1.1,  0.1,  0. ,  0.1,  0.9,  0.7,  1. ,  0.6,  0.3,
         0.1],
       [ 0. ,  0.1,  2.1,  4. ,  3.2,  2.2,  2.4,  2.6,  5.2,  6.8,  7.7,
         9.7],
       [ 0. ,  7.2, 14.4, 21. , 23.8, 30. , 32.1, 34.6, 34.4, 33.1, 32.6,
        32. ],
       [45.5, 43.6, 40.8, 37.4, 24.5, 14.1,  5. ,  0.5,  0. ,  1. ,  2.9,
         3.7]])
```

In [31]:
```
df_sales.describe()
```

Out[31]:

	index	year_id	timeperiod_id	amount	price	benefit_criterion	regret_criterion
count	108.00000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000
mean	269.50000	2008.000000	6.500000	40510.842593	83.215741	9.060185	7.484259
std	31.32092	2.594026	3.468146	9410.871487	16.361741	11.196887	10.887781
min	216.00000	2004.000000	1.000000	22124.000000	65.000000	0.000000	0.000000
25%	242.75000	2006.000000	3.750000	35927.250000	72.675000	0.775000	0.875000
50%	269.50000	2008.000000	6.500000	40867.000000	78.700000	4.350000	3.900000
75%	296.25000	2010.000000	9.250000	47380.500000	83.700000	14.175000	7.950000
max	323.00000	2012.000000	12.000000	60272.000000	127.700000	45.500000	45.500000

In [32]:
```
df_sales['price'].values.reshape(9,12)
```

Out[32]:
```
array([[ 66.8,  70.3,  72.3,  72.8,  72. ,  68.9,  68.2,  68.3,  68.3,
         69.5,  68.9,  69. ],
       [ 72.5,  73.1,  73.3,  72.5,  72.2,  72.1,  72.3,  72.4,  72.4,
         72.4,  72.4,  72.6],
       [ 72.7,  72.9,  74.5,  78.5,  79.6,  80.7,  81.2,  81.1,  83.2,
         83.6,  83.7,  83.8],
       [ 83.7,  83.8,  83.8,  84.1,  84. ,  81.7,  76.2,  73.1,  69.7,
         68.2,  66.6,  65. ],
       [ 65.8,  68.8,  74.3,  75.7,  77.6,  79.4,  81.8,  82.7,  82.7,
         82.5,  82.3,  82.1],
       [ 81. ,  77. ,  77.3,  76.3,  76.2,  76.3,  77.1,  76.9,  77.2,
         76.8,  76.5,  76.3],
       [ 76.4,  76.5,  78.5,  80.4,  79.6,  78.6,  78.8,  79. ,  81.6,
         83.2,  84.1,  86.1],
       [ 93.1, 100.3, 107.5, 114.1, 116.9, 123.1, 125.2, 127.7, 127.5,
        126.2, 125.7, 125.1],
       [124.8, 122.9, 120.1, 116.7, 103.8,  93.4,  84.3,  79.8,  79.3,
         80.3,  82.2,  83. ]])
```

In [33]:
```
df_sales['regret_criterion'].values.reshape(9,12)
```

Out[33]:
```
array([[ 6. ,  2.5,  0.5,  0. ,  0.8,  3.9,  4.6,  4.5,  4.5,  3.3,  3.9,
         3.8],
       [ 0.8,  0.2,  0. ,  0.8,  1.1,  1.2,  1. ,  0.9,  0.9,  0.9,  0.9,
         0.7],
       [11.1, 10.9,  9.3,  5.3,  4.2,  3.1,  2.6,  2.7,  0.6,  0.2,  0.1,
         0. ],
       [ 0.4,  0.3,  0.3,  0. ,  0.1,  2.4,  7.9, 11. , 14.4, 15.9, 17.5,
        19.1],
       [16.9, 13.9,  8.4,  7. ,  5.1,  3.3,  0.9,  0. ,  0. ,  0.2,  0.4,
         0.6],
       [ 0. ,  4. ,  3.7,  4.7,  4.8,  4.7,  3.9,  4.1,  3.8,  4.2,  4.5,
         4.7],
       [ 9.7,  9.6,  7.6,  5.7,  6.5,  7.5,  7.3,  7.1,  4.5,  2.9,  2. ,
         0. ],
       [34.6, 27.4, 20.2, 13.6, 10.8,  4.6,  2.5,  0. ,  0.2,  1.5,  2. ,
         2.6],
       [ 0. ,  1.9,  4.7,  8.1, 21. , 31.4, 40.5, 45. , 45.5, 44.5, 42.6,
        41.8]])
```

In [34]:
```
df_sales.to_csv('df_final_first_3.csv')
```

In [35]:
```
str = '2000 1.101 1.127 1.354 1.407 1.399 1.321 1.378 1.400 1.336 1.308 1.217 1.245 2001 1.140 1.098 1.018 1.002 1
```

In [36]:
```
split_list = str.split(' ')
```

In [37]:
```
year_dict = {}
current_year = 2000
for val in split_list:
    int_val = float(val)
    if int_val >= current_year:
        year_dict[int_val] = []
        current_year = int_val
    else:
        year_dict[current_year].append(int_val)
```

In [38]:
```
float_list = []
for i in split_list:
    float_val = float(i)
    if float_val < 1999:
        float_list.append(float(i))
```

In [39]:
```
np.array(float_list).reshape(8,12)
```

Out[39]:
```
array([[1.101, 1.127, 1.354, 1.407, 1.399, 1.321, 1.378, 1.4  , 1.336,
        1.308, 1.217, 1.245],
       [1.14 , 1.098, 1.018, 1.002, 1.028, 1.108, 1.062, 0.997, 0.924,
        0.87 , 0.953, 0.98 ],
       [1.04 , 1.118, 1.245, 1.293, 1.284, 1.164, 1.032, 1.097, 1.041,
        1.002, 1.041, 1.077],
       [0.912, 0.945, 1.018, 1.325, 1.623, 1.71 , 1.512, 1.538, 1.507,
        1.369, 1.344, 1.182],
       [1.189, 1.279, 1.602, 1.733, 1.565, 1.24 , 1.391, 1.915, 1.632,
        1.542, 1.374, 1.126],
       [1.313, 1.234, 1.429, 1.484, 1.632, 1.751, 2.105, 2.108, 1.742,
        1.394, 1.084, 1.385],
       [1.164, 1.418, 1.776, 1.867, 1.349, 1.132, 1.138, 1.473, 1.595,
        1.517, 1.49 , 1.366],
       [1.483, 1.962, 1.995, 2.327, 2.574, 2.49 , 2.49 , 2.455, 2.618,
        2.449, 1.857, 1.601]])
```

In [40]:
```
df_inventory[df_inventory['year_id'] > 2003]
```

Out[40]:

	commodity_desc	attribute_desc	unit_desc	year_id	timeperiod_id	amount
1966	Catfish-Broodfish	Producer Inventories	1,000 EA	2004	17	1113.0
1967	Catfish-Broodfish	Producer Inventories	1,000 EA	2005	17	1053.0
1968	Catfish-Broodfish	Producer Inventories	1,000 EA	2006	17	1091.0
1969	Catfish-Broodfish	Producer Inventories	1,000 EA	2007	17	886.0
1970	Catfish-Broodfish	Producer Inventories	1,000 EA	2008	17	801.0
...	...	...	...	...	...	...
2099	Catfish-Large Food-size	Producer Inventories	1,000 EA	2012	17	3595.0
2100	Catfish-Large Food-size	Producer Inventories	1,000 EA	2013	17	5155.0
2101	Catfish-Large Food-size	Producer Inventories	1,000 EA	2014	17	4500.0
2102	Catfish-Large Food-size	Producer Inventories	1,000 EA	2015	17	5090.0
2103	Catfish-Large Food-size	Producer Inventories	1,000 EA	2016	17	3520.0

78 rows × 6 columns

In [41]:
```
mean_year = df_sales_his[['year_id','price']].groupby('year_id').mean().values
```

In [42]:
```
df_sales_his['datetime'] = pd.date_range(start='1/1/1986', periods=326, freq='M')
```

In [43]:
```
initial_year = 1986
for i in mean_year:
    df_sales_his.loc[df_sales_his['year_id'] == initial_year, 'percent_diff'] = (df_sales_his['price'] - i[0])/i[0
    initial_year+=1
```

In [61]:
```
min_val_his = df_sales_his[['timeperiod_id', 'percent_diff']].groupby('timeperiod_id').min().values
max_val_his = df_sales_his[['timeperiod_id', 'percent_diff']].groupby('timeperiod_id').max().values
```

In [45]:
```
for i in range(12):
    df_sales.loc[df_sales['timeperiod_id'] == i+1, 'lower_bound'] = df_sales['price'] * (1 + min_val_his[i])
    df_sales.loc[df_sales['timeperiod_id'] == i+1, 'upper_bound'] = df_sales['price'] * (1 + max_val_his[i])
```

In [46]:
```
min_val_lower = df_sales[['year_id', 'lower_bound']].groupby('year_id').min().values
max_val_lower = df_sales[['year_id', 'lower_bound']].groupby('year_id').max().values

min_val_upper = df_sales[['year_id', 'upper_bound']].groupby('year_id').min().values
max_val_upper = df_sales[['year_id', 'upper_bound']].groupby('year_id').max().values
```

In [47]:
```
calculate_benefit_val(df_sales, min_val_lower, 'benefit_criterion_lower', 2003, 'lower_bound')
calculate_benefit_val(df_sales, min_val_upper, 'benefit_criterion_upper', 2003, 'upper_bound')

calculate_criterion_val(df_sales, max_val_lower, 'regret_criterion_lower', 2003, 'lower_bound')
calculate_criterion_val(df_sales, max_val_upper, 'regret_criterion_upper', 2003, 'upper_bound')
```

In [48]:
```
df_sales
```

Out[48]:

datetime	index	year_id	timeperiod_id	commodity_desc	amount	price	benefit_criterion	regret_criterion	lower_bound	upper_bound	b
2004-01-31	216	2004	1	Catfish	53849.0	66.8	0.0	6.0	52.838403	85.460174	
2004-02-29	217	2004	2	Catfish	54173.0	70.3	3.5	2.5	59.907307	88.568631	
2004-03-31	218	2004	3	Catfish	60272.0	72.3	5.5	0.5	66.034410	89.013121	
2004-04-30	219	2004	4	Catfish	53896.0	72.8	6.0	0.0	70.573322	87.091338	
2004-05-31	220	2004	5	Catfish	52324.0	72.0	5.2	0.8	70.665213	80.345013	
...	...	...	...	...	...	...	...	...	...	...	
2012-08-31	319	2012	8	Catfish	24886.0	79.8	0.5	45.0	65.279754	86.579949	
2012-09-30	320	2012	9	Catfish	24535.0	79.3	0.0	45.5	64.464275	85.902719	
2012-10-31	321	2012	10	Catfish	28596.0	80.3	1.0	44.5	66.100359	86.099065	
2012-11-30	322	2012	11	Catfish	22124.0	82.2	2.9	42.6	69.265402	87.787086	
2012-12-31	323	2012	12	Catfish	22653.0	83.0	3.7	41.8	69.733157	89.831374	

108 rows × 14 columns

In [49]:
```
lower_matrix = df_sales['lower_bound'].values.reshape(9,12)
benefit_lower_matrix = df_sales['benefit_criterion_lower'].values.reshape(9,12)
regret_lower_matrix = df_sales['regret_criterion_lower'].values.reshape(9,12)
```

In [50]:
```
upper_matrix = df_sales['upper_bound'].values.reshape(9,12)
benefit_upper_matrix = df_sales['benefit_criterion_upper'].values.reshape(9,12)
regret_upper_matrix = df_sales['regret_criterion_upper'].values.reshape(9,12)
```

In [ ]:
```

```

In [52]:
```
df_sales.to_csv('df_model_matrix.csv')
```

In [62]:
```
print(min_val_his)
print(max_val_his)
```

```
[[-0.20900595]
 [-0.14783347]
 [-0.086661  ]
 [-0.03058624]
 [-0.01853871]
 [-0.04254229]
 [-0.13582778]
 [-0.18195797]
 [-0.18708355]
 [-0.17683239]
 [-0.1573552 ]
 [-0.15984148]]
[[0.27934393]
 [0.25986674]
 [0.23116351]
 [0.19630958]
 [0.11590296]
 [0.06576802]
 [0.06372133]
 [0.08496177]
 [0.08326253]
 [0.0722175 ]
 [0.06796941]
 [0.08230571]]
```

In [ ]: