gression-on-premature-baby-dataset

June 16, 2023

[ ]:
```
# using Binary logistic regression for predicting any missing values in a␣
 ↪dataset


# logistic regression predicts something is true or false instead of predicting␣
 ↪something continuous
# it is a special case of linear regression
# it is also called as sigmoid function
# sigmoid function = 1/(1+e^-value)

import pandas as pd                 #importing all the necessary libraries
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report , accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sns


data_df = pd.read_csv('/content/preterm.csv')  # reading the path of the dataset
data_df.head()       # displaying the parameters of the dataset

data_df.columns       # displaying the number of columns of dataset
data_df[['Count Contraction', 'lenght of contraction', 'STD', 'Entropy',
       'Contraction times', 'Pre-term']]
```

[ ]:

	Count Contraction	lenght of contraction	STD	Entropy	\
0	11055	218320	53231.010	1.860	
1	9118	222820	62367.488	1.580	
2	7925	13481	60503.050	2.067	
3	12451	17474	53628.078	1.731	
4	11152	218320	53317.910	1.857	

5	6029	63781	59177.965	1.701
6	10052	22310	54431.030	1.790
7	9101	219830	63467.583	1.490
8	7929	13192	61503.160	2.067
9	12452	16473	54678.091	1.701
10	11121	228321	52901.880	1.777
11	6179	65782	62177.988	1.991
12	2012	12483	32901.880	0.901
13	1812	10489	30905.820	1.210
14	1919	11297	31902.320	1.320
15	2012	12483	32901.880	0.901
16	1812	11481	29205.840	1.310
17	1829	11296	31902.320	1.020
18	2119	12298	32103.350	1.120
19	714	6185	57003.089	0.701
20	694	10226	40605.810	0.821
21	614	10143	49114.330	0.832
22	674	6185	42902.890	0.901
23	615	6388	49406.860	0.881
24	625	6351	41912.340	0.892
25	428	6283	39206.370	0.618
26	435	5308	47003.089	0.501
27	428	3388	39103.320	0.524
28	545	3308	39104.320	0.537
29	448	3315	39206.370	0.504
30	495	3328	47003.089	0.581
31	415	3355	41103.320	0.592
32	495	3348	42104.320	0.519
33	525	3608	49206.370	0.529
34	415	3309	47003.089	0.539
35	465	3238	52103.140	0.524
36	425	2308	51104.320	0.591
37	501	2308	54206.370	0.598
38	412	2308	51003.089	0.509
39	415	2308	59103.320	0.498
40	465	2358	49104.320	0.488
41	425	2333	52206.370	0.475
42	380	2334	57003.089	0.479
43	323	2339	49103.320	0.498
44	222	2675	59104.320	0.477
45	321	2456	59206.370	0.458
46	334	2682	57003.089	0.445
47	321	2901	49103.320	0.469
48	345	2932	51104.320	0.488
49	320	2963	43102.120	0.457
50	349	2333	58013.079	0.428
51	395	2334	49208.120	0.445

52	398	2339	51124.340	0.469
53	321	2675	46107.090	0.499
54	398	2339	51122.310	0.469
55	321	2675	46108.180	0.498
56	398	2336	51224.370	0.459
57	323	2641	46102.170	0.439

	Contraction times	Pre-term
0	2	1
1	2	1
2	2	1
3	2	1
4	2	1
5	2	1
6	2	1
7	2	1
8	2	1
9	2	1
10	2	1
11	2	1
12	1	1
13	1	1
14	1	1
15	1	1
16	1	1
17	1	1
18	1	1
19	1	0
20	1	0
21	1	0
22	1	0
23	1	0
24	1	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
30	0	0
31	0	0
32	0	0
33	0	0
34	0	0
35	0	0
36	0	0
37	0	0
38	0	0

```
39                  0         0
40                  0         0
41                  0         0
42                  0         0
43                  0         0
44                  0         0
45                  0         0
46                  0         0
47                  0         0
48                  0         0
49                  0         0
50                  0         0
51                  0         0
52                  0         0
53                  0         0
54                  0         0
55                  0         0
56                  0         0
57                  0         0
```

Diagram representing the flow of type of birth prediction in babies

[ ]:
```
data_df.isna()      # any missing values
# or data_df.isna().any()
```

[ ]:
```
    Count Contraction  lenght of contraction    STD  Entropy  \
0               False                  False  False    False
1               False                  False  False    False
2               False                  False  False    False
3               False                  False  False    False
4               False                  False  False    False
5               False                  False  False    False
6               False                  False  False    False
7               False                  False  False    False
8               False                  False  False    False
9               False                  False  False    False
10              False                  False  False    False
11              False                  False  False    False
12              False                  False  False    False
13              False                  False  False    False
14              False                  False  False    False
15              False                  False  False    False
16              False                  False  False    False
17              False                  False  False    False
18              False                  False  False    False
19              False                  False  False    False
20              False                  False  False    False
```

21	False	False	False	False
22	False	False	False	False
23	False	False	False	False
24	False	False	False	False
25	False	False	False	False
26	False	False	False	False
27	False	False	False	False
28	False	False	False	False
29	False	False	False	False
30	False	False	False	False
31	False	False	False	False
32	False	False	False	False
33	False	False	False	False
34	False	False	False	False
35	False	False	False	False
36	False	False	False	False
37	False	False	False	False
38	False	False	False	False
39	False	False	False	False
40	False	False	False	False
41	False	False	False	False
42	False	False	False	False
43	False	False	False	False
44	False	False	False	False
45	False	False	False	False
46	False	False	False	False
47	False	False	False	False
48	False	False	False	False
49	False	False	False	False
50	False	False	False	False
51	False	False	False	False
52	False	False	False	False
53	False	False	False	False
54	False	False	False	False
55	False	False	False	False
56	False	False	False	False
57	False	False	False	False

	Contraction times	Pre-term
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False

```
8               False     False
9               False     False
10              False     False
11              False     False
12              False     False
13              False     False
14              False     False
15              False     False
16              False     False
17              False     False
18              False     False
19              False     False
20              False     False
21              False     False
22              False     False
23              False     False
24              False     False
25              False     False
26              False     False
27              False     False
28              False     False
29              False     False
30              False     False
31              False     False
32              False     False
33              False     False
34              False     False
35              False     False
36              False     False
37              False     False
38              False     False
39              False     False
40              False     False
41              False     False
42              False     False
43              False     False
44              False     False
45              False     False
46              False     False
47              False     False
48              False     False
49              False     False
50              False     False
51              False     False
52              False     False
53              False     False
54              False     False
```

```
55              False     False
56              False     False
57              False     False
```

[ ]:
```
data_df[['Count Contraction', 'lenght of contraction', 'STD', 'Entropy',
       'Contraction times', 'Pre-term']].describe()

# The describe() method returns description of the data in the DataFrame.

# If the DataFrame contains numerical data, the description contains these␣
 ↪information for each column:

# count - The number of not-empty values.
# mean - The average (mean) value.
# std - The standard deviation.
# min - the minimum value.
# 25% - The 25% percentile*.
# 50% - The 50% percentile*.
# 75% - The 75% percentile*.
# max - the maximum value.

# *Percentile meaning: how many of the values are less than the given␣
 ↪percentile.
```

[ ]:
```
       Count Contraction  lenght of contraction           STD    Entropy  \
count          58.000000              58.000000     58.000000  58.000000
mean         2503.810345           26621.965517  48564.968190   0.879759
std          3788.639864           61527.769917   8952.845551   0.528180
min           222.000000            2308.000000  29205.840000   0.428000
25%           398.000000            2649.500000  42303.962500   0.490500
50%           495.000000            3371.500000  49307.490000   0.586000
75%          1988.750000           12093.750000  54374.865000   1.187500
max         12452.000000          228321.000000  63467.583000   2.067000

       Contraction times   Pre-term
count          58.000000  58.000000
mean            0.637931   0.327586
std             0.809988   0.473432
min             0.000000   0.000000
25%             0.000000   0.000000
50%             0.000000   0.000000
75%             1.000000   1.000000
max             2.000000   1.000000
```

[ ]:
```
data_df.columns       # displaying the number of columns of dataset
data_df[['Count Contraction', 'lenght of contraction', 'STD', 'Entropy',
       'Contraction times', 'Pre-term']]
```

[ ]:

	Count Contraction	lenght of contraction	STD	Entropy	\
0	11055	218320	53231.010	1.860	
1	9118	222820	62367.488	1.580	
2	7925	13481	60503.050	2.067	
3	12451	17474	53628.078	1.731	
4	11152	218320	53317.910	1.857	
5	6029	63781	59177.965	1.701	
6	10052	22310	54431.030	1.790	
7	9101	219830	63467.583	1.490	
8	7929	13192	61503.160	2.067	
9	12452	16473	54678.091	1.701	
10	11121	228321	52901.880	1.777	
11	6179	65782	62177.988	1.991	
12	2012	12483	32901.880	0.901	
13	1812	10489	30905.820	1.210	
14	1919	11297	31902.320	1.320	
15	2012	12483	32901.880	0.901	
16	1812	11481	29205.840	1.310	
17	1829	11296	31902.320	1.020	
18	2119	12298	32103.350	1.120	
19	714	6185	57003.089	0.701	
20	694	10226	40605.810	0.821	
21	614	10143	49114.330	0.832	
22	674	6185	42902.890	0.901	
23	615	6388	49406.860	0.881	
24	625	6351	41912.340	0.892	
25	428	6283	39206.370	0.618	
26	435	5308	47003.089	0.501	
27	428	3388	39103.320	0.524	
28	545	3308	39104.320	0.537	
29	448	3315	39206.370	0.504	
30	495	3328	47003.089	0.581	
31	415	3355	41103.320	0.592	
32	495	3348	42104.320	0.519	
33	525	3608	49206.370	0.529	
34	415	3309	47003.089	0.539	
35	465	3238	52103.140	0.524	
36	425	2308	51104.320	0.591	
37	501	2308	54206.370	0.598	
38	412	2308	51003.089	0.509	
39	415	2308	59103.320	0.498	
40	465	2358	49104.320	0.488	
41	425	2333	52206.370	0.475	
42	380	2334	57003.089	0.479	
43	323	2339	49103.320	0.498	
44	222	2675	59104.320	0.477	
45	321	2456	59206.370	0.458	

46	334	2682	57003.089	0.445
47	321	2901	49103.320	0.469
48	345	2932	51104.320	0.488
49	320	2963	43102.120	0.457
50	349	2333	58013.079	0.428
51	395	2334	49208.120	0.445
52	398	2339	51124.340	0.469
53	321	2675	46107.090	0.499
54	398	2339	51122.310	0.469
55	321	2675	46108.180	0.498
56	398	2336	51224.370	0.459
57	323	2641	46102.170	0.439

	Contraction times	Pre-term
0	2	1
1	2	1
2	2	1
3	2	1
4	2	1
5	2	1
6	2	1
7	2	1
8	2	1
9	2	1
10	2	1
11	2	1
12	1	1
13	1	1
14	1	1
15	1	1
16	1	1
17	1	1
18	1	1
19	1	0
20	1	0
21	1	0
22	1	0
23	1	0
24	1	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
30	0	0
31	0	0
32	0	0

```
33                  0         0
34                  0         0
35                  0         0
36                  0         0
37                  0         0
38                  0         0
39                  0         0
40                  0         0
41                  0         0
42                  0         0
43                  0         0
44                  0         0
45                  0         0
46                  0         0
47                  0         0
48                  0         0
49                  0         0
50                  0         0
51                  0         0
52                  0         0
53                  0         0
54                  0         0
55                  0         0
56                  0         0
57                  0         0
```

[ ]:
```
# percentage of total number of not  preterm
npreterm = 0
notpreterm = data_df['Pre-term']
for i in range (len(notpreterm)):
  if notpreterm[i] ==0:
    npreterm = npreterm+1
len(notpreterm)
```

[ ]:
```
58
```

[ ]:
```
per_nf = (npreterm/len(notpreterm))*100
print('percentage of not preterm babies:',per_nf)
```

```
percentage of not preterm babies: 67.24137931034483
```

[ ]:
```
# percentage of total number of  pretterm
pterm = 0
preterm = data_df['Pre-term']
for i in range (len(preterm)):
  if preterm[i] ==1:
     pterm = pterm + 1
```

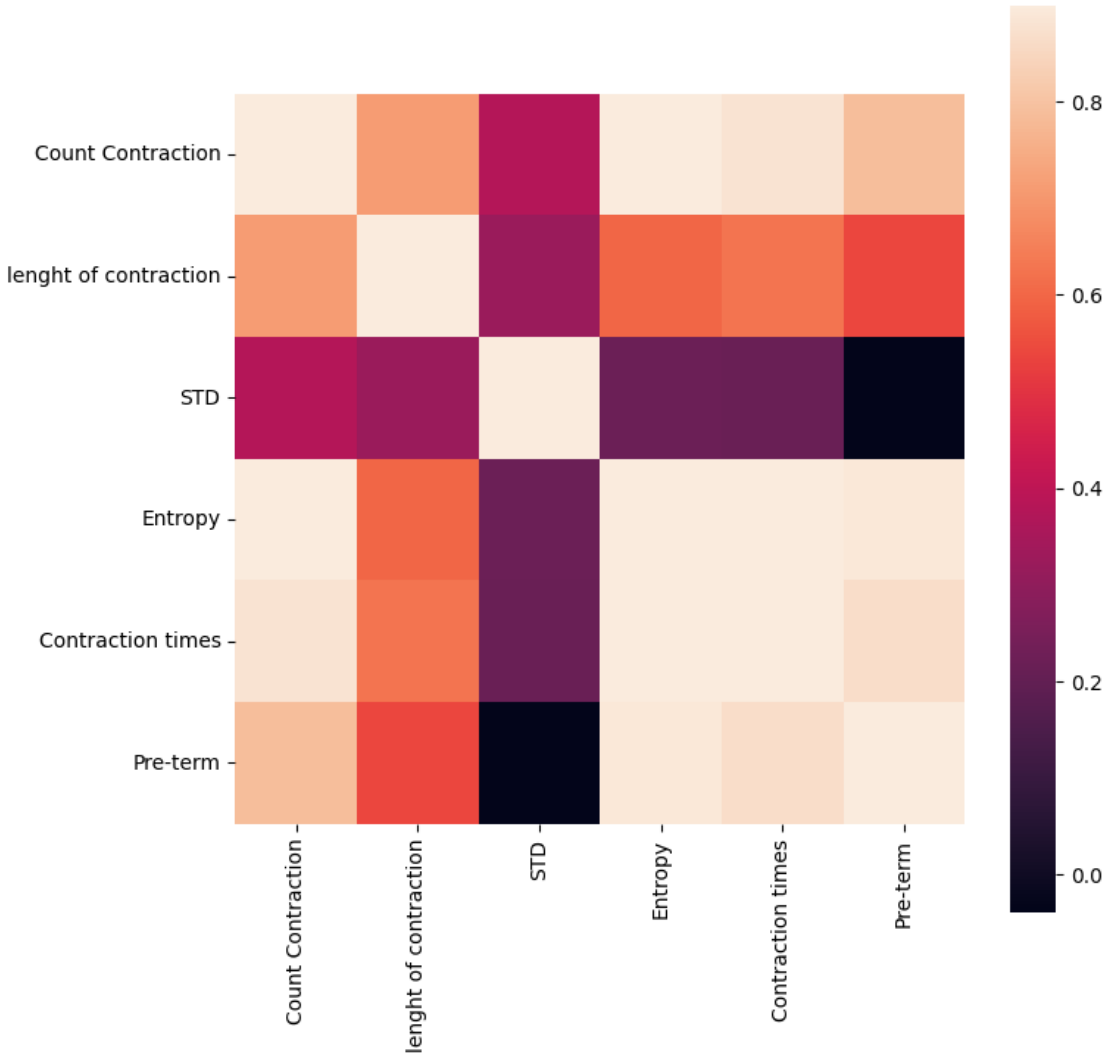
```
len(preterm)
```

[ ]: 58

[ ]:
```
per_nf = (pterm/len(preterm))*100
print('percentage of not preterm babies:',per_nf)
```

```
percentage of not preterm babies: 32.758620689655174
```

[ ]:
```
# correlation matrix
# The measure of the relationship between two variables statistically is called␣
 ↪"Correlation".
# Given a vast amount of observed data, it is hard to determine how closely two␣
 ↪variables are related.
# It is a major need for data science and data analysis.
# Statistical techniques are used to organize all the data to get the␣
 ↪correlation view, and for that,
# graphs and other representations are made.

correlation_metrics = data_df.corr()
fig = plt.figure(figsize = (8,8))
sns.heatmap(correlation_metrics , vmax = 0.9 , square = True)
plt.show()

# A correlation matrix is simply a table which displays the correlation␣
 ↪coefficients for different variables.
# The matrix depicts the correlation between all the possible pairs of values␣
 ↪in a table.
# It is a powerful tool to summarize a large dataset and to identify and␣
 ↪visualize patterns in the given data
```



[ ]:
```
x = data_df.drop(['Contraction times'] , axis = 1)
y = data_df['Contraction times']
xtrain , xtest , ytrain , ytest = train_test_split(x,y, test_size = 0.2 ,␣
 ↪random_state =42)
xtest.shape
logisticreg = LogisticRegression()
logisticreg.fit(xtrain , ytrain)


# x_train : The training part of the first sequence ( x )
# x_test : The test part of the first sequence ( x )
# y_train : The training part of the second sequence ( y )
# y_test : The test part of the second sequence ( y )
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
y_pred = logisticreg.predict(xtest)
logisticsreg = LogisticRegression()
y_pred
```

[ ]:
```
array([2, 2, 0, 1, 0, 0, 0, 1, 0, 1, 0, 2])
```

[ ]:
```
accuracy = logisticreg.score(xtest , ytest)
cm = metrics.confusion_matrix(ytest , y_pred)
print(cm)
```

```
[[6 1 0]
 [0 2 0]
 [0 0 3]]
```

[ ]:
```
print('accuracy score of the logistic regression model for contraction times is␣
  ↪:', accuracy*100,'%')
```

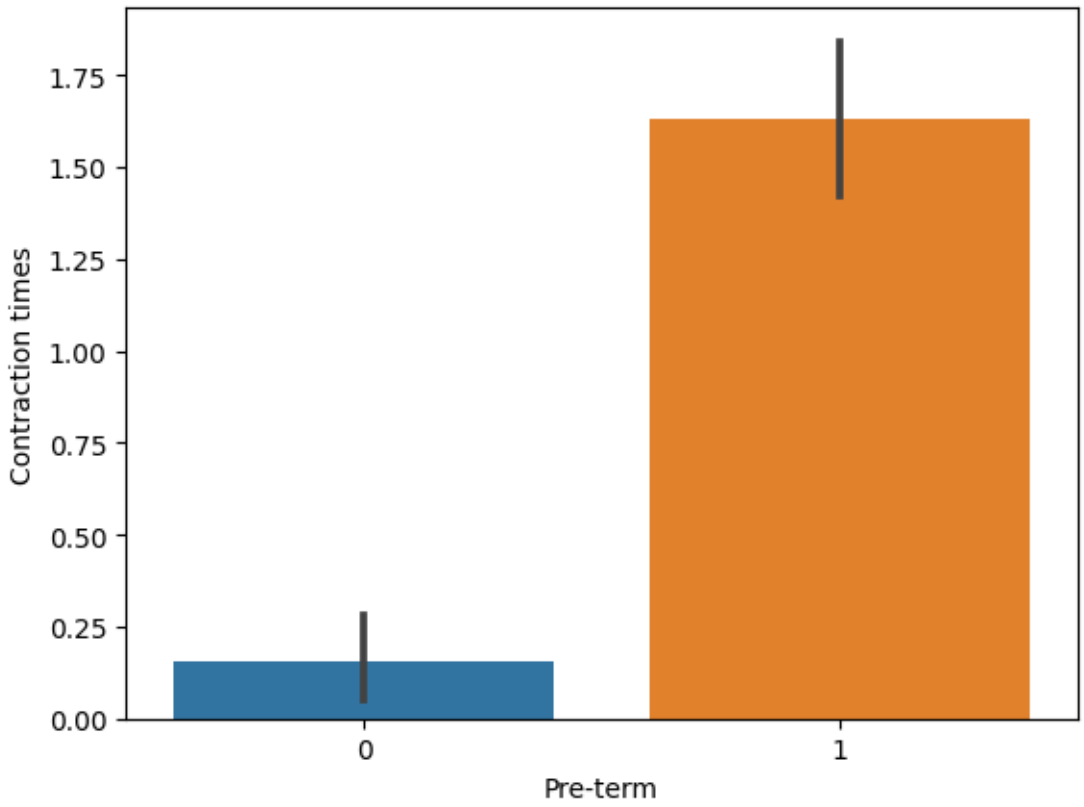
```
accuracy score of the logistic regression model is : 91.66666666666666 %
```

[ ]:
```
x = data_df.drop(['Pre-term'] , axis = 1)
y = data_df['Pre-term']
xtrain , xtest , ytrain , ytest = train_test_split(x,y, test_size = 0.2 ,␣
  ↪random_state =42)
xtest.shape
logisticreg = LogisticRegression()
logisticreg.fit(xtrain , ytrain)

y_pred = logisticreg.predict(xtest)
logisticsreg = LogisticRegression()
y_pred

accuracy = logisticreg.score(xtest , ytest)
cm = metrics.confusion_matrix(ytest , y_pred)
print(cm)
```

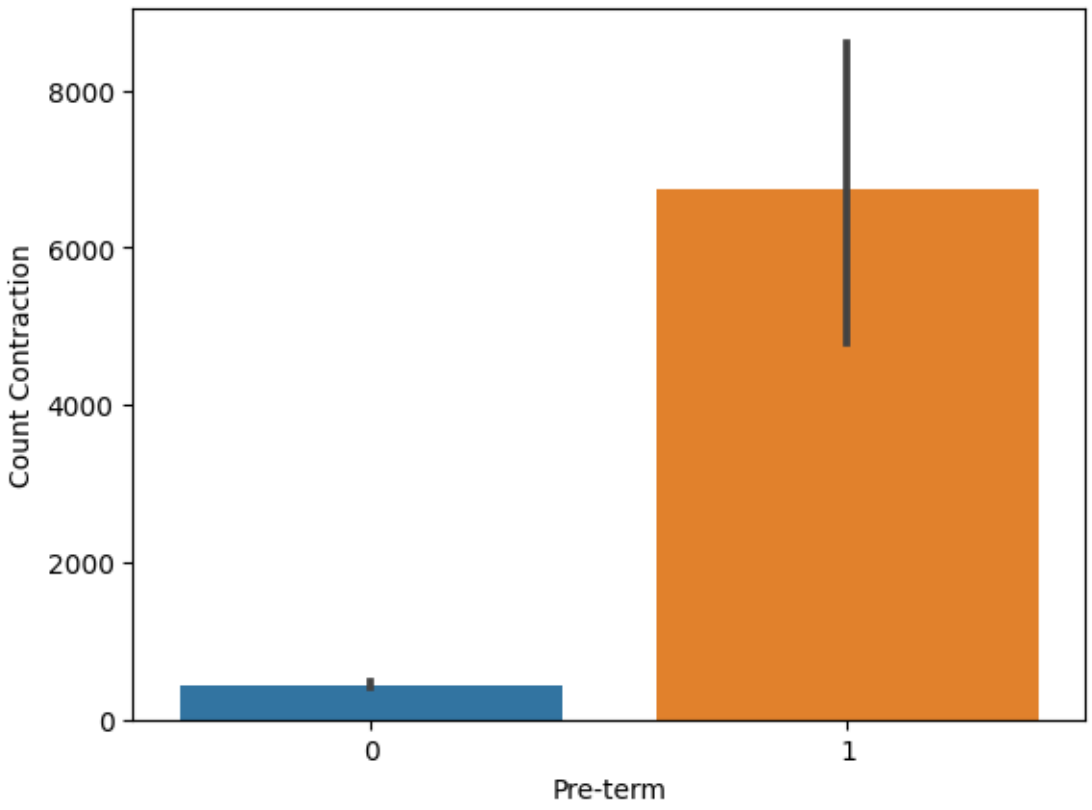
```
print('accuracy score of the logistic regression modelfor preterm is :',␣
 ↪accuracy*100,'%')
```

```
[[7 0]
 [0 5]]
accuracy score of the logistic regression model is : 100.0 %
```

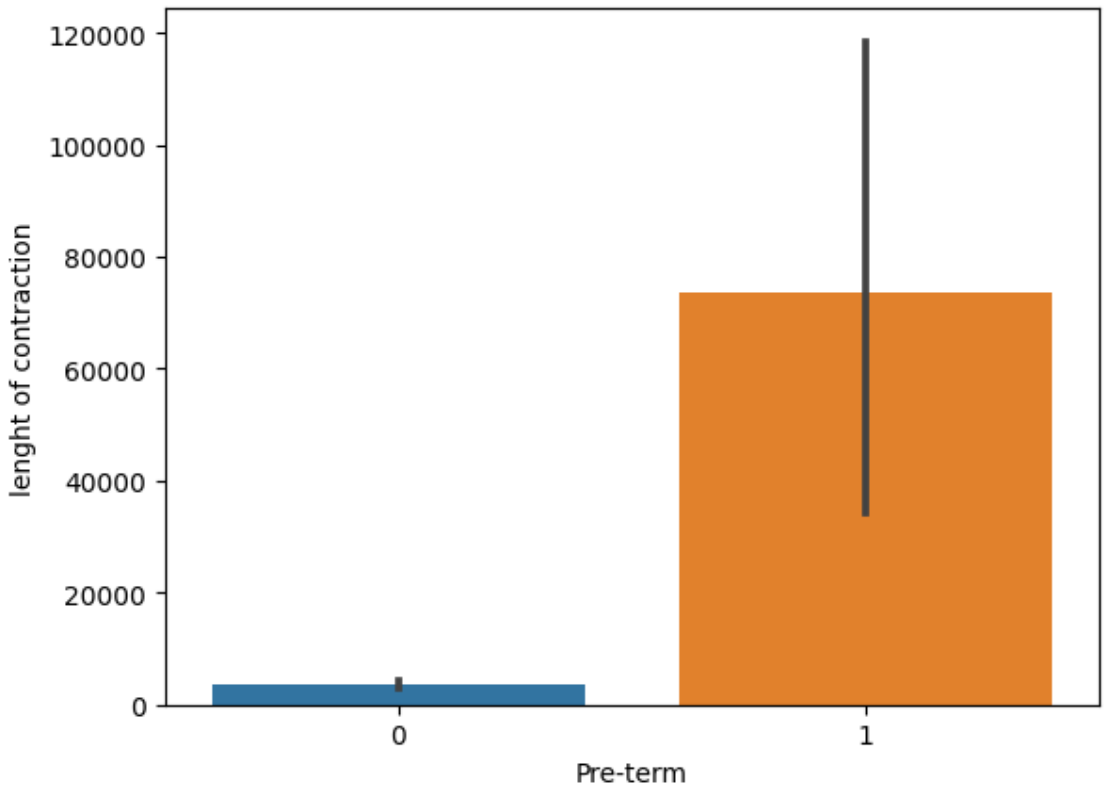
[ ]:
```
sns.barplot(x = 'Pre-term',y = 'Contraction times' , data = data_df)
plt.show()
```



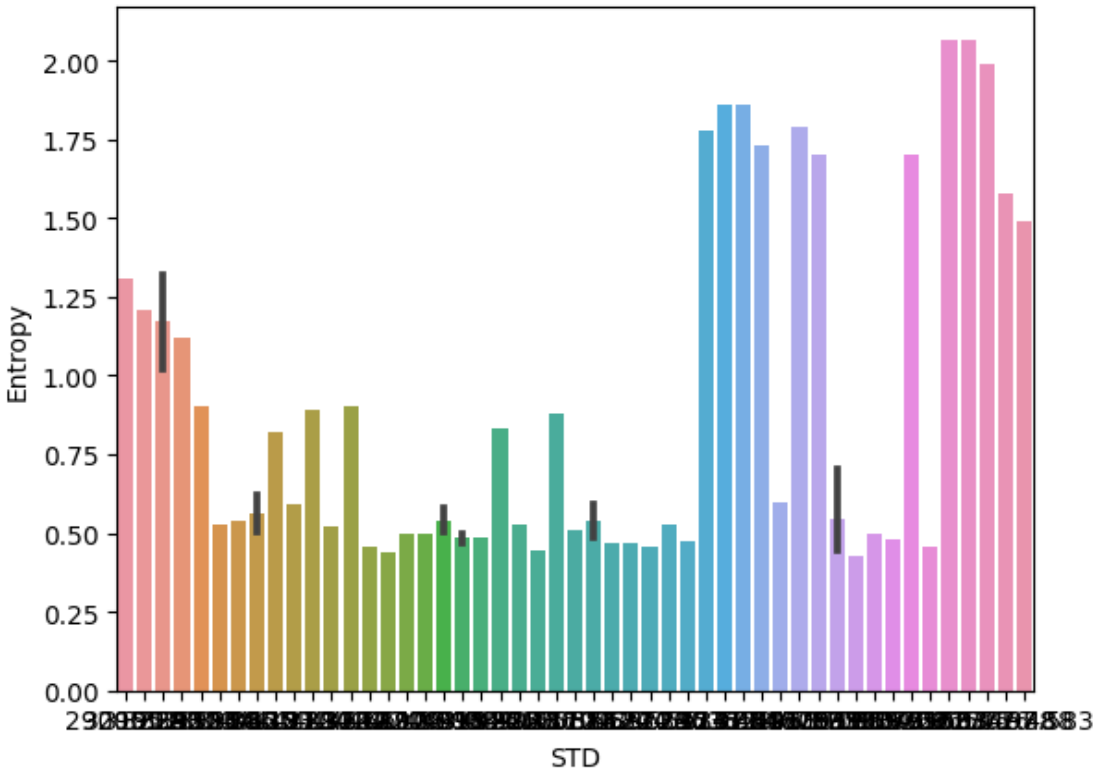
[ ]:
```
sns.barplot(x = 'Pre-term',y = 'Count Contraction' , data = data_df)
plt.show()
```



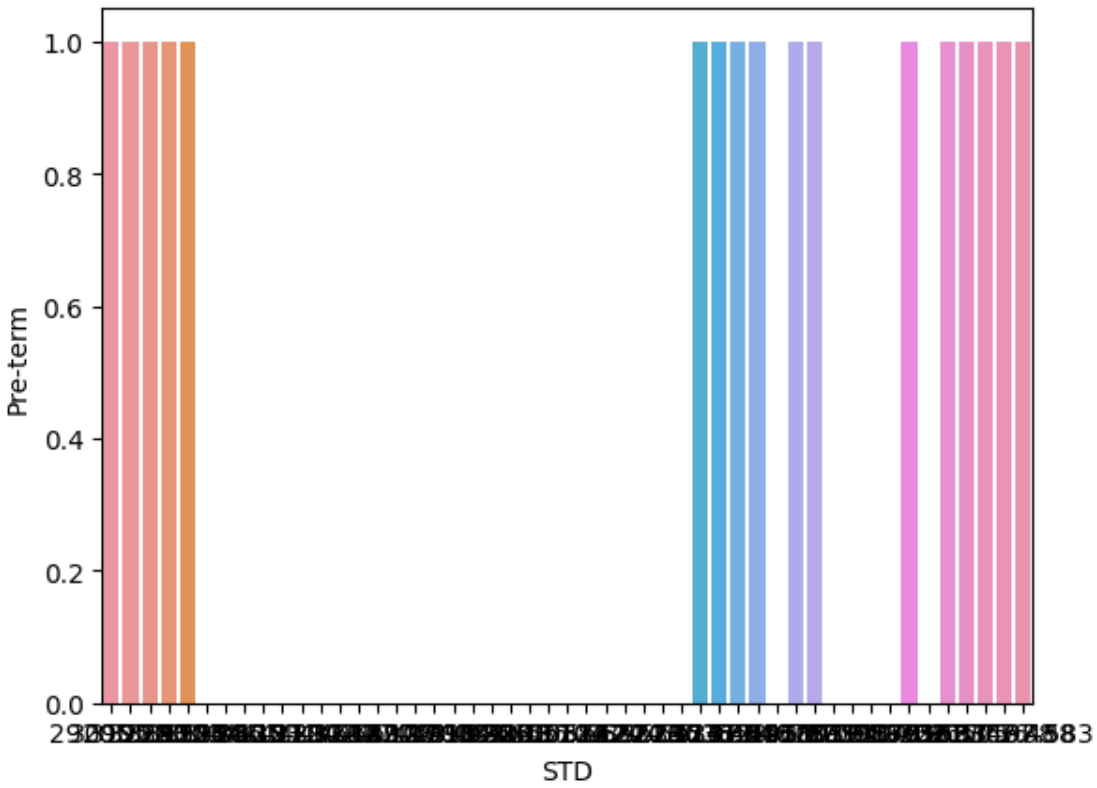
[ ]:
```
sns.barplot(x = 'Pre-term',y = 'lenght of contraction' , data = data_df)
plt.show()
```



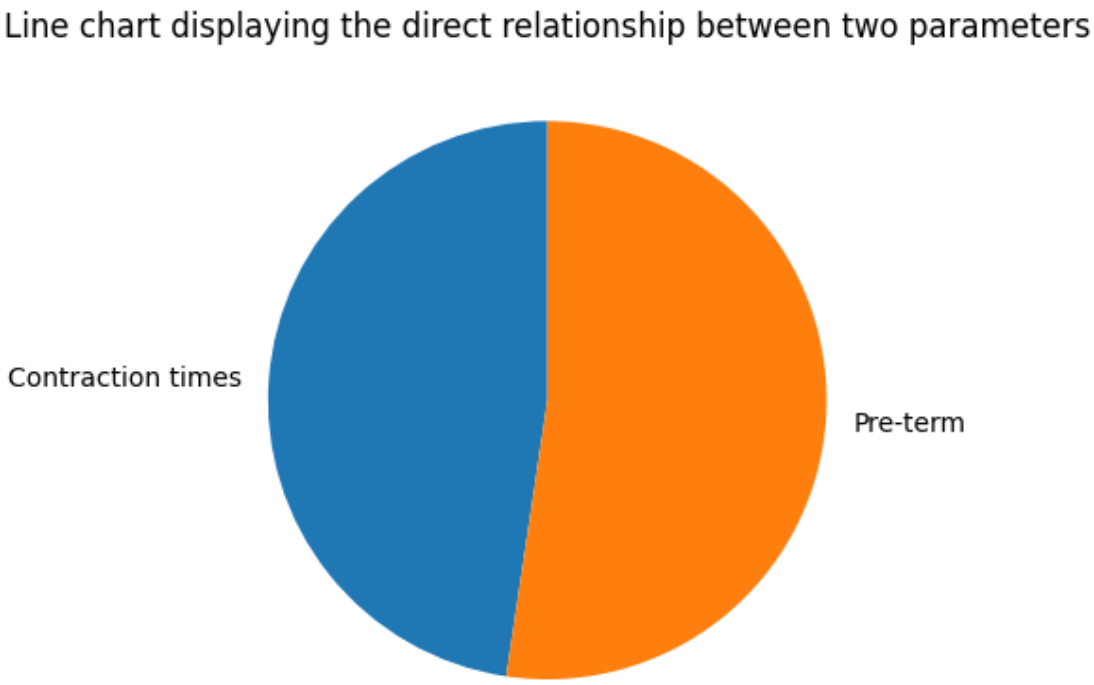
[ ]:
```
sns.barplot(x = 'STD',y = 'Entropy' , data = data_df)
plt.show()
```



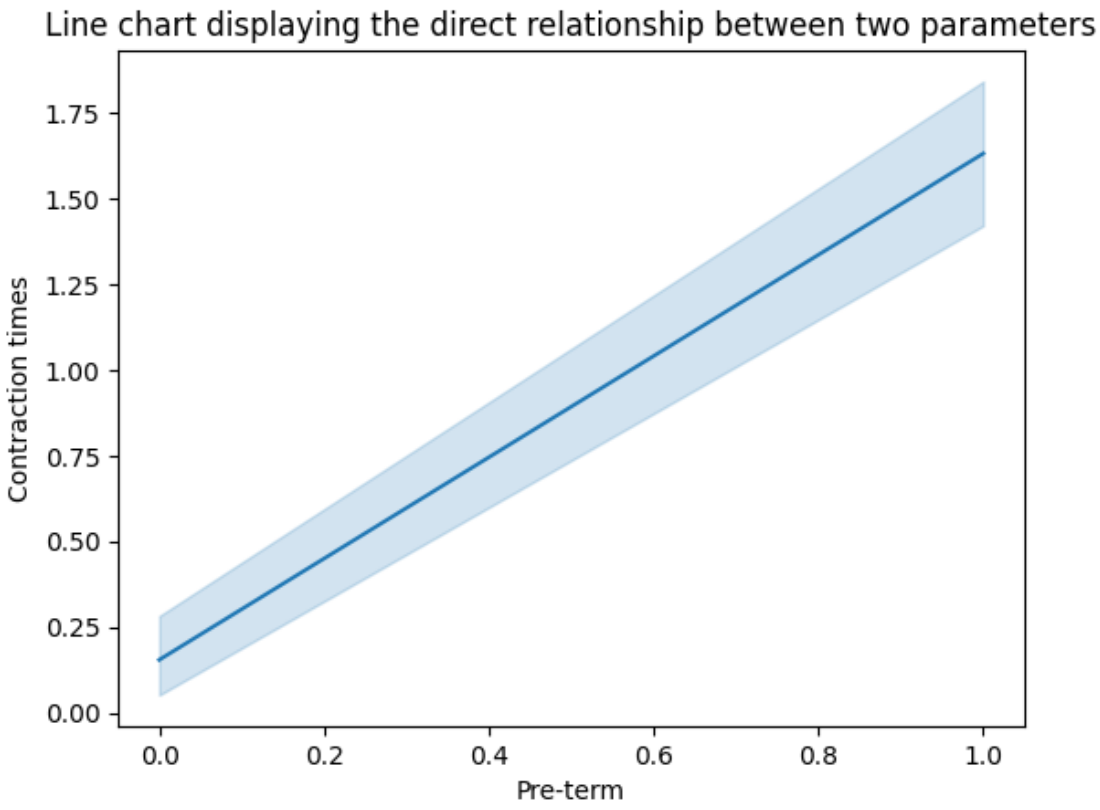
[ ]:
```
sns.barplot(x = 'STD',y = 'Pre-term' , data = data_df)
plt.show()
```



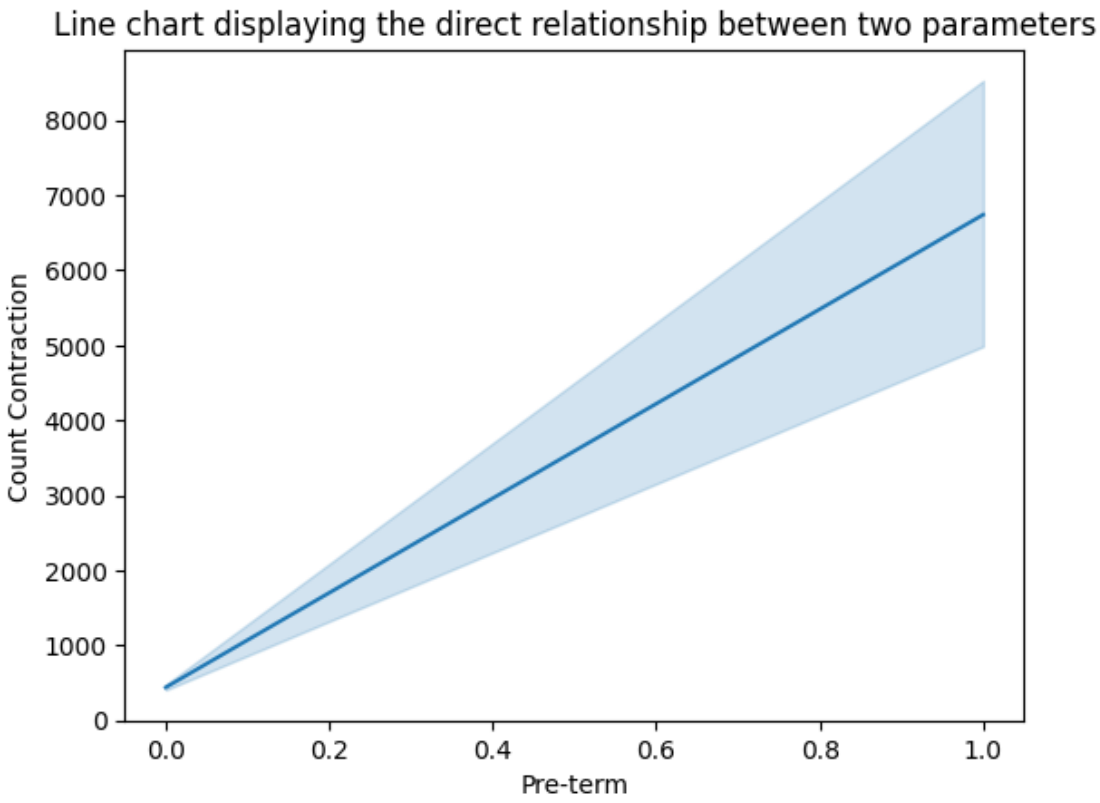
[ ]:
```
#from os import terminal_size
#data_df[['Count Contraction', 'lenght of contraction', 'STD', 'Entropy',
#       'Contraction times', 'Pre-term']]

y = np.array([91, 100])
mylabels = ['Contraction times', 'Pre-term']

#myexplode = [0.2, 0]

plt.pie( y ,  labels = mylabels ,  startangle = 90 ) # , explode = myexplode ,␣
 ↪shadow = True)
#fig = plt.figure(figsize =(10, 7))
plt.title("Line chart displaying the direct relationship between two␣
 ↪parameters")
# show plot
plt.show()
```



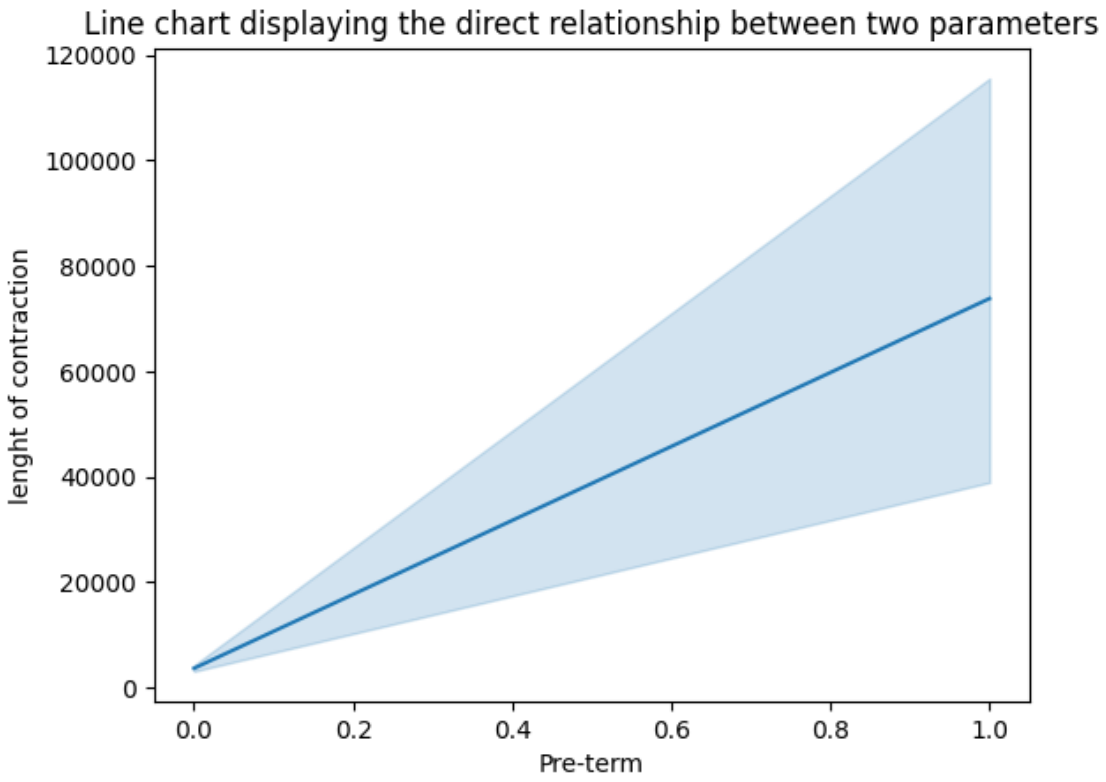
[ ]:
```
#lineplot
import seaborn as sns
sns.lineplot(x = 'Pre-term',y = 'Contraction times',data = data_df)
plt.title("Line chart displaying the direct relationship between two␣
 ↪parameters")
plt.show()
```



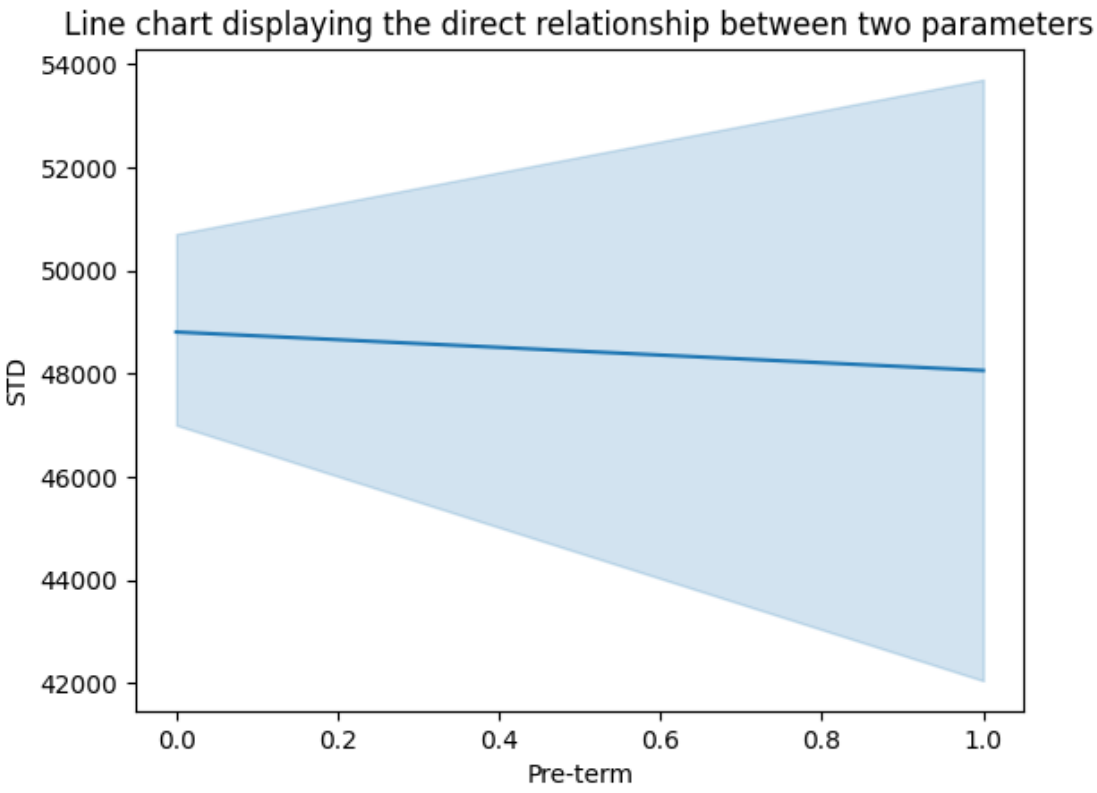
[ ]:
```
#lineplot
import seaborn as sns
sns.lineplot(x = 'Pre-term',y = 'Count Contraction',data = data_df)
plt.title("Line chart displaying the direct relationship between two␣
  ↪parameters")
plt.show()
```



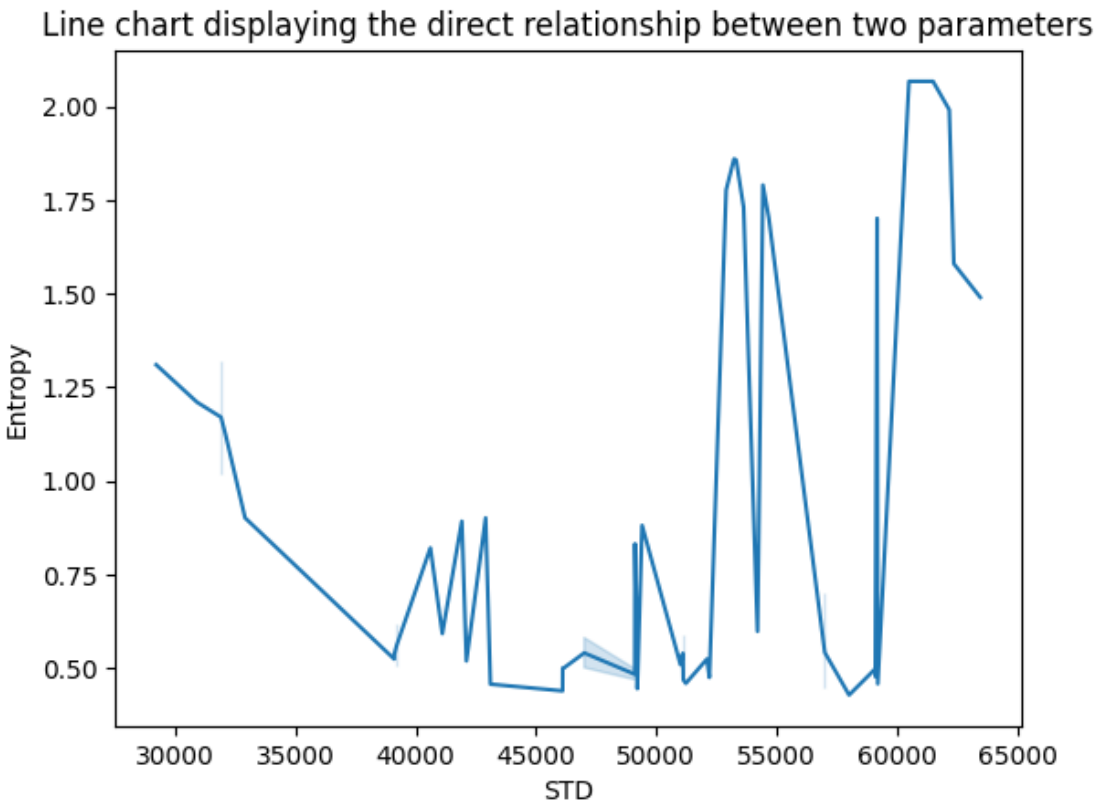
[ ]:
```
#lineplot
import seaborn as sns
sns.lineplot(x = 'Pre-term',y = 'lenght of contraction',data = data_df)
plt.title("Line chart displaying the direct relationship between two␣
 ↪parameters")
plt.show()
```



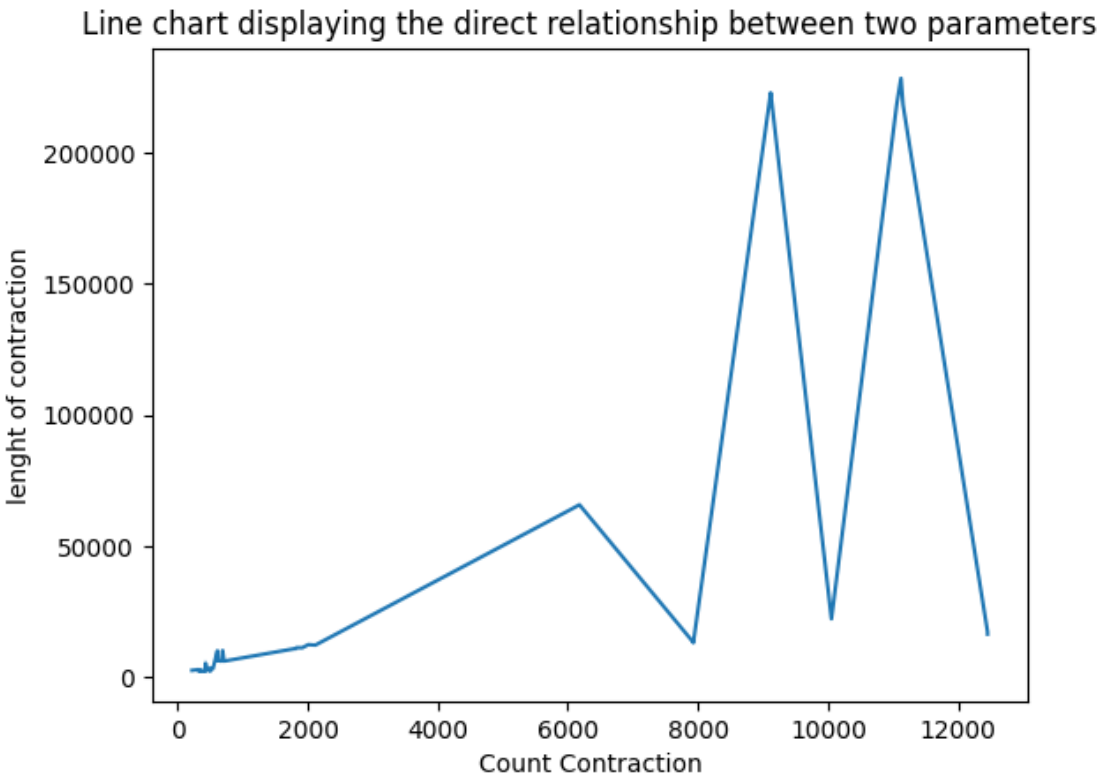
[ ]:
```
#lineplot
import seaborn as sns
sns.lineplot(x = 'Pre-term',y = 'STD',data = data_df)
plt.title("Line chart displaying the direct relationship between two␣
 ↪parameters")
plt.show()
```



[ ]:
```
#lineplot
import seaborn as sns
sns.lineplot(x = 'STD',y = 'Entropy',data = data_df)
plt.title("Line chart displaying the direct relationship between two␣
 ↪parameters")
plt.show()
```



[ ]:
```
#lineplot
import seaborn as sns
sns.lineplot(x = 'Count Contraction',y = 'lenght of contraction',data = data_df)
plt.title("Line chart displaying the direct relationship between two␣
  ↪parameters")
plt.show()
```


Line chart displaying the direct relationship between two parameters
lenght of contraction
200000
150000
100000
50000
0
0
2000
4000
6000
8000
10000
12000
Count Contraction