In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('preterm.csv')
```

In [3]:

```
df.head()
```

Out[3]:

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

In [4]:

```
df.tail()
```

Out[4]:

	Count Contraction	lenght of contraction	STD	Entropy	Contraction times	Pre-term
53	321	2675	46107.09	0.499	0	0
54	398	2339	51122.31	0.469	0	0
55	321	2675	46108.18	0.498	0	0
56	398	2336	51224.37	0.459	0	0
57	323	2641	46102.17	0.439	0	0

In [5]:

```
df.shape
```

Out[5]:

```
(58, 6)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Count Contraction', 'lenght of contraction', 'STD', 'Entropy',
       'Contraction times', 'Pre-term'],
      dtype='object')
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
1
```

In [8]:

```
df = df.drop_duplicates()
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
Count Contraction        0
lenght of contraction    0
STD                      0
Entropy                  0
Contraction times        0
Pre-term                 0
dtype: int64
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 57 entries, 0 to 57
Data columns (total 6 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Count Contraction      57 non-null     int64  
 1   lenght of contraction  57 non-null     int64  
 2   STD                    57 non-null     float64
 3   Entropy                57 non-null     float64
 4   Contraction times      57 non-null     int64  
 5   Pre-term               57 non-null     int64  
dtypes: float64(2), int64(4)
memory usage: 3.1 KB
```

In [11]:

```
df.describe()
```

Out[11]:

	Count Contraction	lenght of contraction	STD	Entropy	Contraction times	Pre-term
count	57.000000	57.000000	57.000000	57.000000	57.000000	57.000000
mean	2512.438596	26870.017544	48839.759211	0.879386	0.631579	0.315789
std	3821.742366	62045.428533	8782.195737	0.532868	0.815729	0.468961
min	222.000000	2308.000000	29205.840000	0.428000	0.000000	0.000000
25%	398.000000	2641.000000	42902.890000	0.488000	0.000000	0.000000
50%	495.000000	3355.000000	49406.860000	0.581000	0.000000	0.000000
75%	1919.000000	11481.000000	54431.030000	1.210000	1.000000	1.000000
max	12452.000000	228321.000000	63467.583000	2.067000	2.000000	1.000000

In [12]:

```
df.nunique()
```

Out[12]:

```
Count Contraction        44
lenght of contraction    46
STD                      49
Entropy                  47
Contraction times         3
Pre-term                  2
dtype: int64
```

In [13]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [15]:

```
df['Contraction times'].unique()
```

Out[15]:

```
array([2, 1, 0], dtype=int64)
```

In [16]:

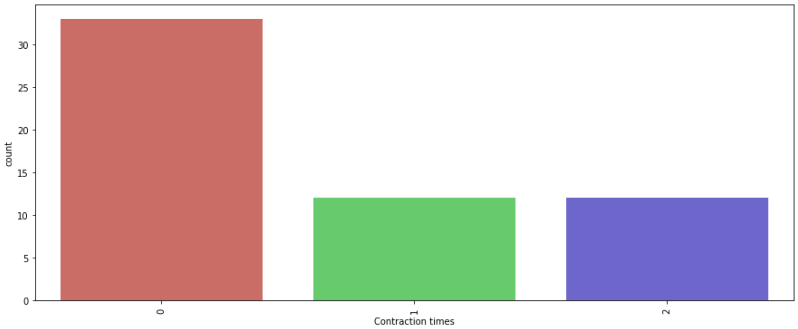
```
df['Contraction times'].value_counts()
```

Out[16]:

```
0    33
2    12
1    12
Name: Contraction times, dtype: int64
```

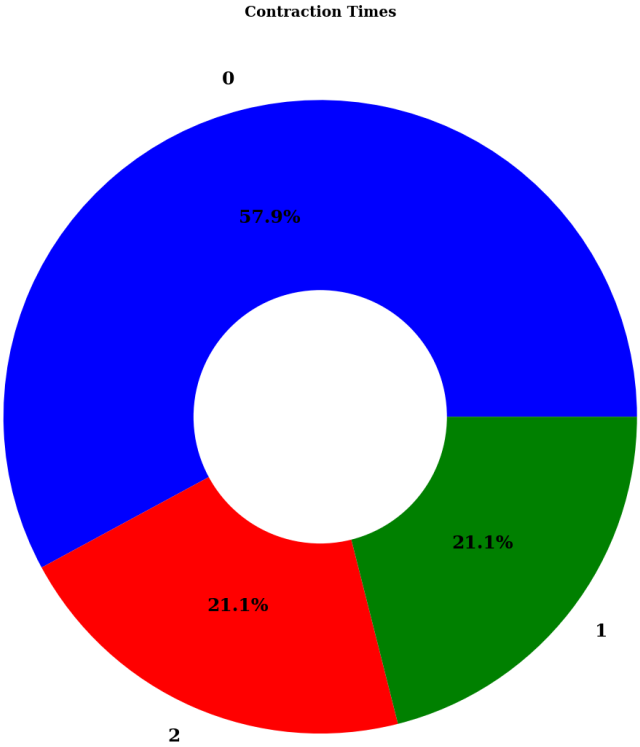
In [17]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['Contraction times'], data = df,
              palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [18]:

```
label_data = df['Contraction times'].value_counts()

explode = (0.0, 0.0, 0.0)
plt.figure(figsize=(30, 20))
patches, texts, pcts = plt.pie(label_data,
                               labels = label_data.index,
                               colors = ['blue', 'red', 'green'],
                               pctdistance = 0.65,
                               shadow = False,
                               startangle = 0,
                               explode = explode,
                               autopct = '%1.1f%%',
                               textprops={ 'fontsize': 25,
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
plt.setp(pcts, color='black')

hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Contraction Times', size=20, **hfont)

centre_circle = plt.Circle((0,0),0.40,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.show()
```



In [19]:

```
df['Pre-term'].unique()
```

Out[19]:

```
array([1, 0], dtype=int64)
```

In [20]:

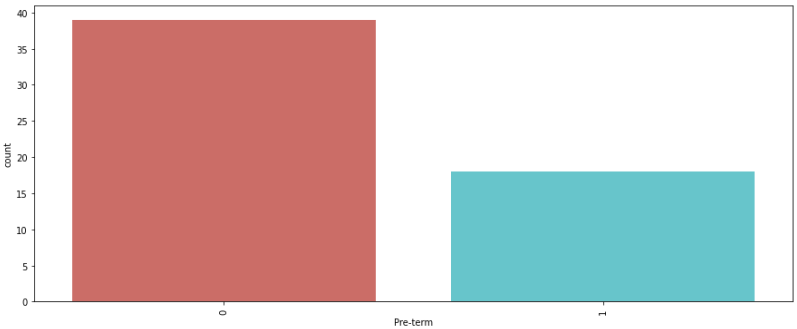
```
df['Pre-term'].value_counts()
```

Out[20]:

```
0    39
1    18
Name: Pre-term, dtype: int64
```

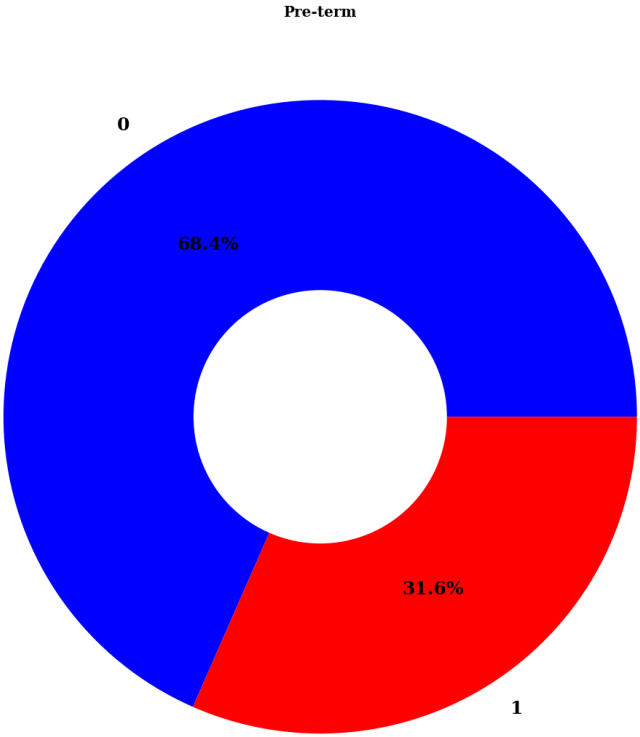
In [21]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['Pre-term'], data = df,
              palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



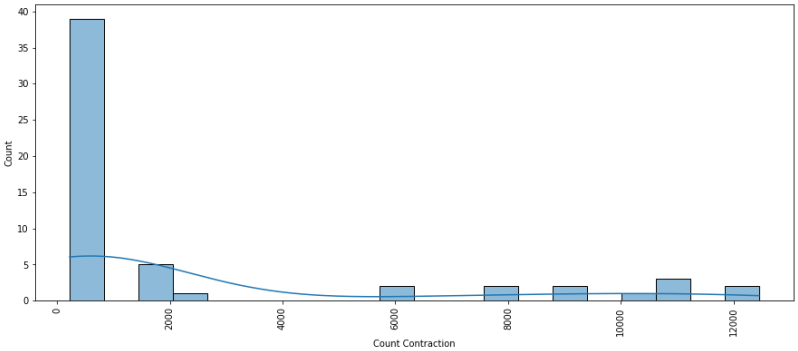
In [22]:

```
label_data = df['Pre-term'].value_counts()

explode = (0.0, 0.0)
plt.figure(figsize=(30, 20))
patches, texts, pcts = plt.pie(label_data,
                               labels = label_data.index,
                               colors = ['blue', 'red'],
                               pctdistance = 0.65,
                               shadow = False,
                               startangle = 0,
                               explode = explode,
                               autopct = '%1.1f%%',
                               textprops={ 'fontsize': 25,
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
plt.setp(pcts, color='black')

hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Pre-term', size=20, **hfont)

centre_circle = plt.Circle((0,0),0.40,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.show()
```



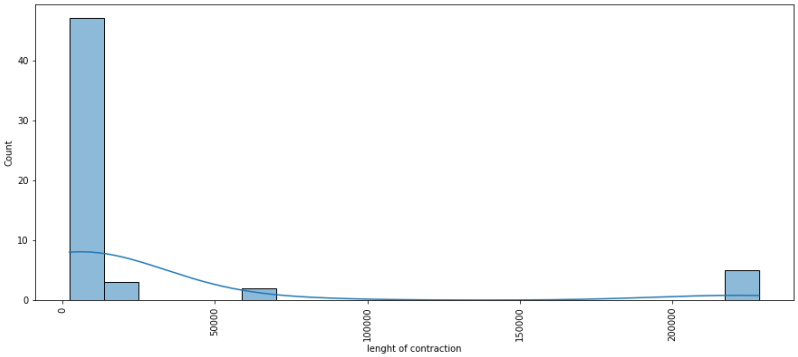
In [23]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['Count Contraction'], bins = 20, kde = True,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



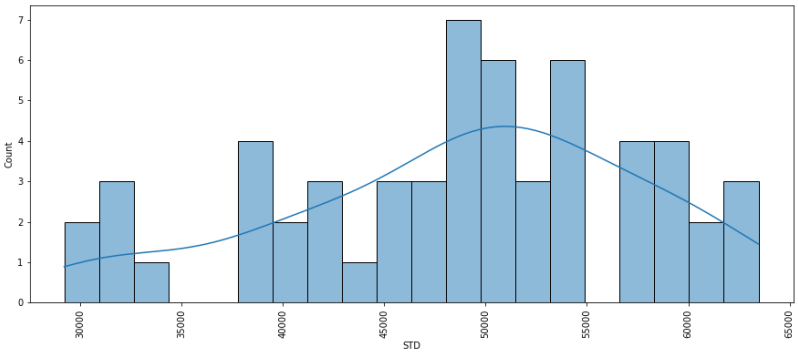
In [24]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['lenght of contraction'], bins = 20, kde = True,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



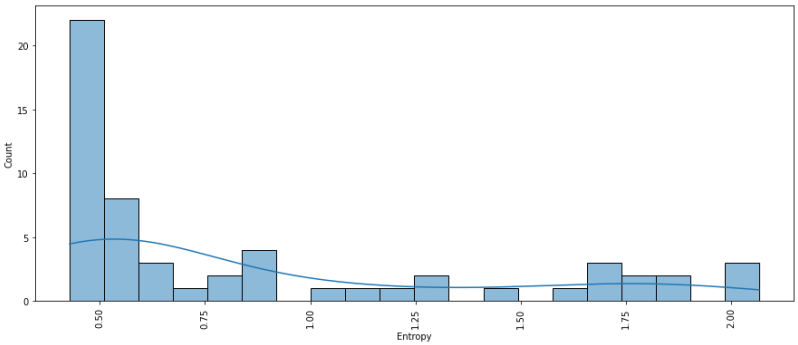
In [25]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['STD'], bins = 20, kde = True,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



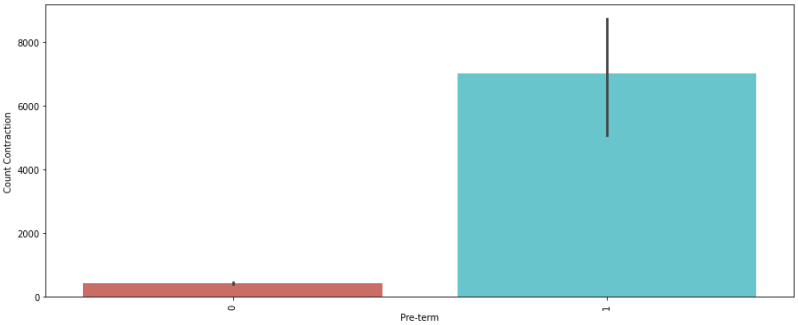
In [26]:

```
plt.figure(figsize=(15,6))
sns.histplot(df['Entropy'], bins = 20, kde = True,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



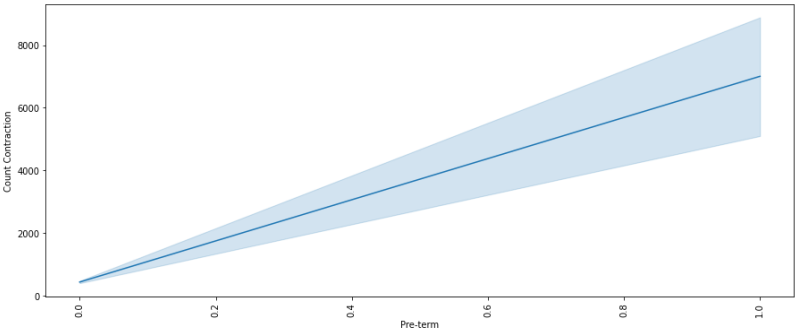
In [27]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['Count Contraction'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



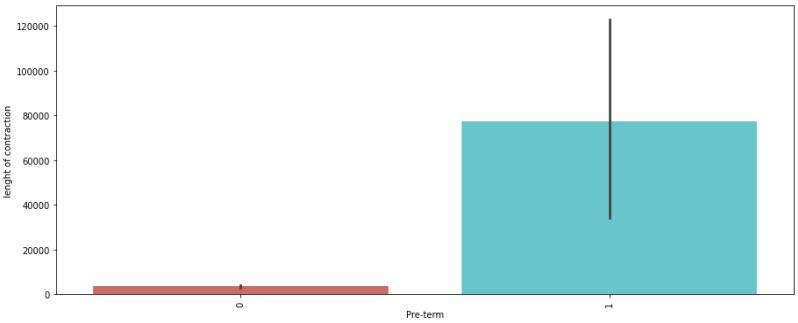
In [28]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['Count Contraction'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



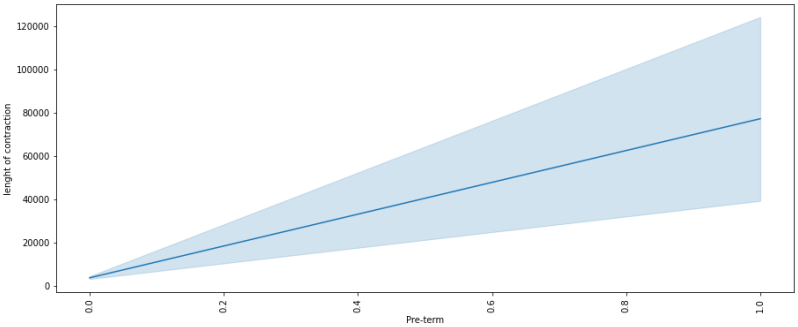
In [29]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['lenght of contraction'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



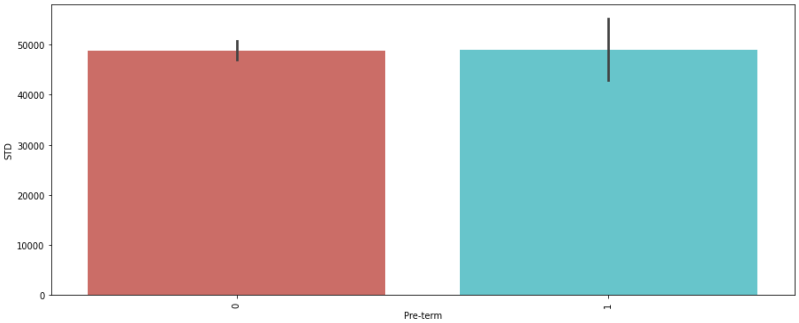
In [30]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['lenght of contraction'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



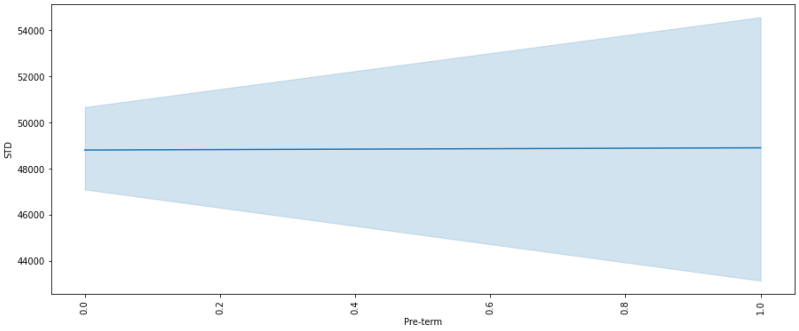
In [31]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['STD'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



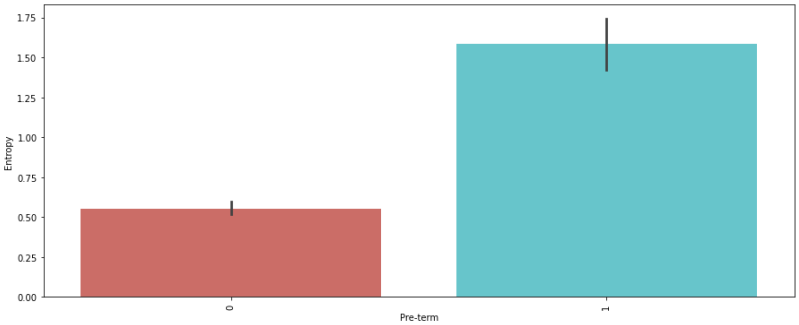
In [32]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['STD'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



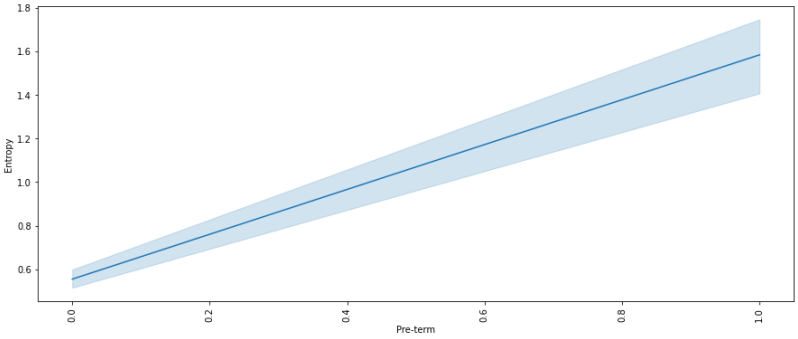
In [33]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['Entropy'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



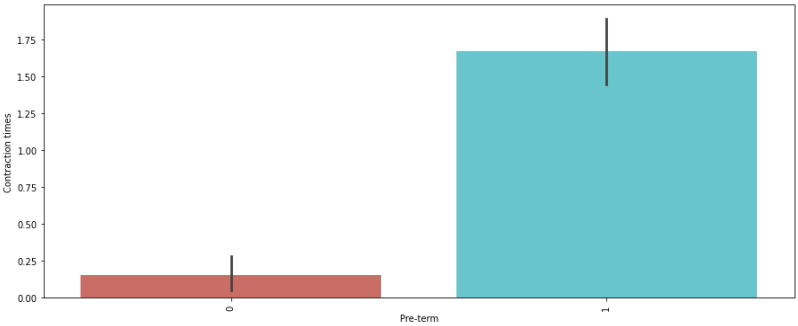
In [34]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['Entropy'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



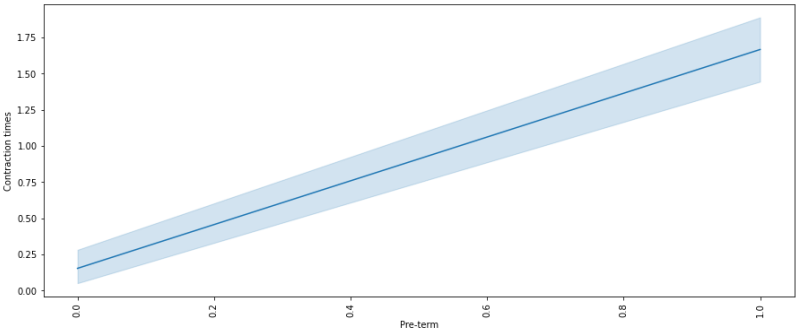
In [35]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = df['Contraction times'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



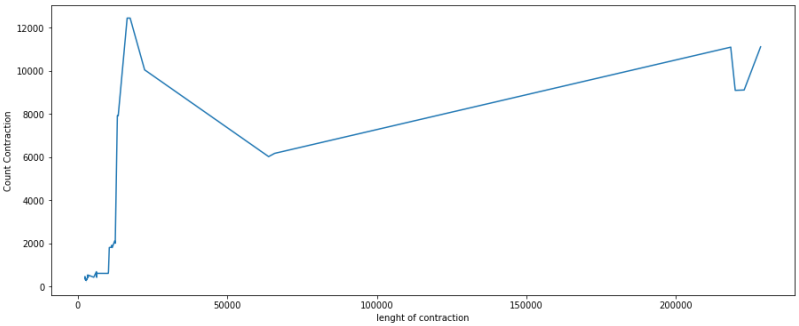
In [36]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['Contraction times'], x = df['Pre-term'], data = df,
             palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



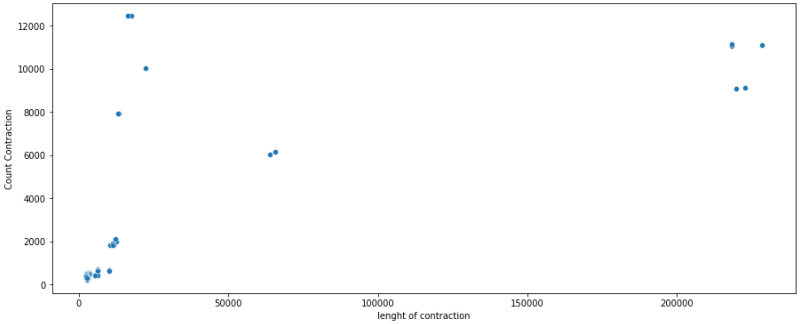
In [37]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['Count Contraction'], x = df['lenght of contraction'], data = df,
             palette = 'hls')
plt.show()
```



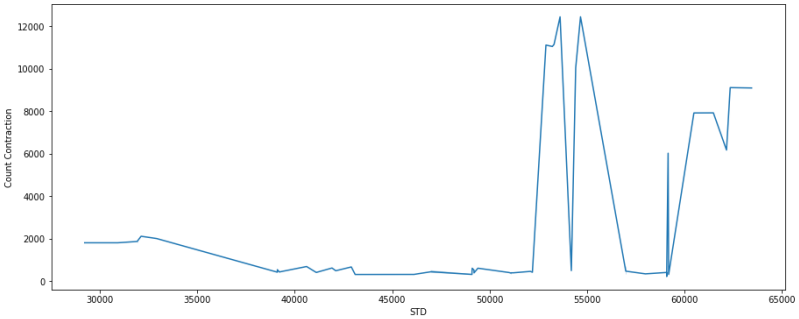
In [38]:

```
plt.figure(figsize=(15,6))
sns.scatterplot(y = df['Count Contraction'], x = df['lenght of contraction'], data = df,
                palette = 'hls')
plt.show()
```



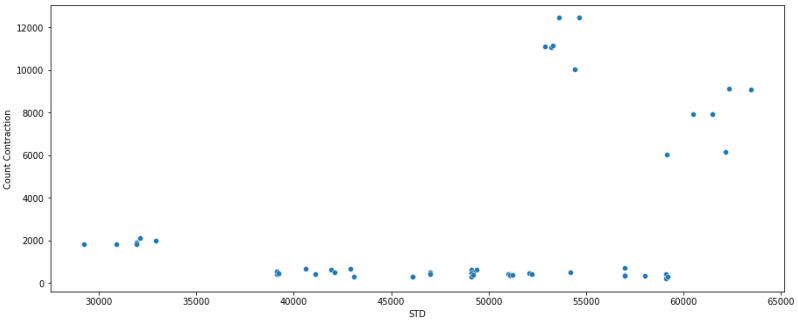
In [39]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['Count Contraction'], x = df['STD'], data = df,
             palette = 'hls')
plt.show()
```



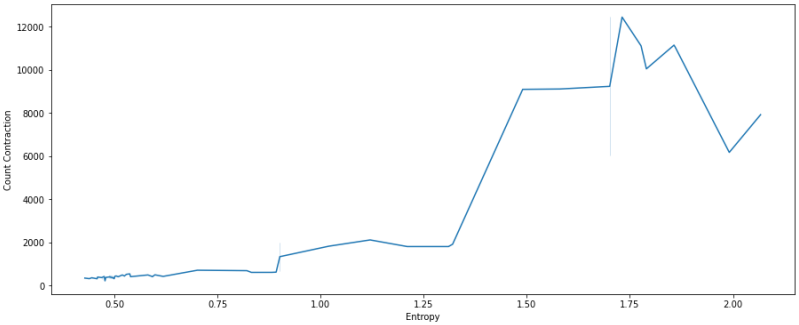
In [40]:

```
plt.figure(figsize=(15,6))
sns.scatterplot(y = df['Count Contraction'], x = df['STD'], data = df,
                palette = 'hls')
plt.show()
```



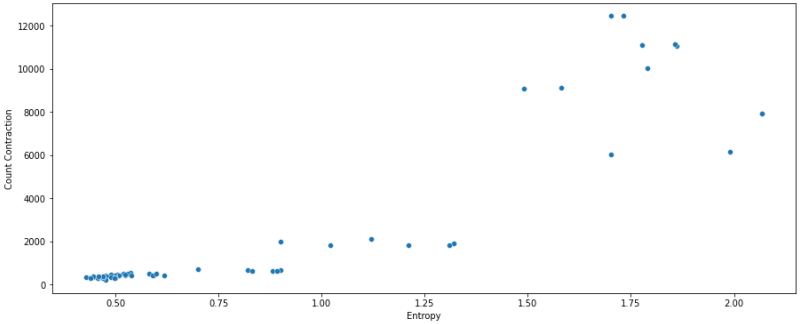
In [41]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['Count Contraction'], x = df['Entropy'], data = df,
             palette = 'hls')
plt.show()
```



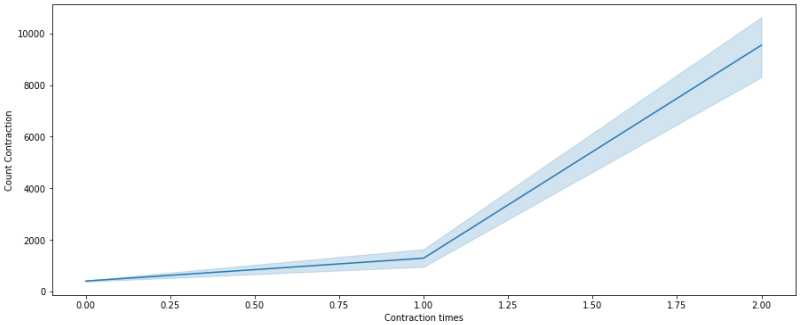
In [42]:

```
plt.figure(figsize=(15,6))
sns.scatterplot(y = df['Count Contraction'], x = df['Entropy'], data = df,
                palette = 'hls')
plt.show()
```



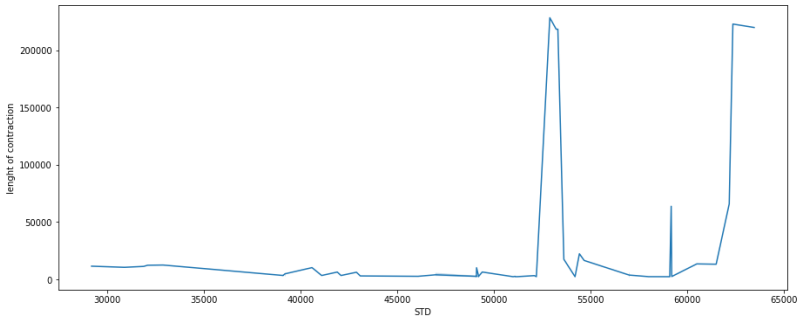
In [43]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['Count Contraction'], x = df['Contraction times'], data = df,
             palette = 'hls')
plt.show()
```



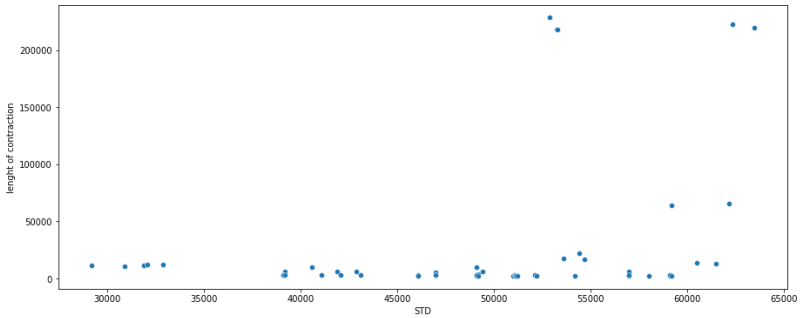
In [44]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['lenght of contraction'], x = df['STD'], data = df,
             palette = 'hls')
plt.show()
```



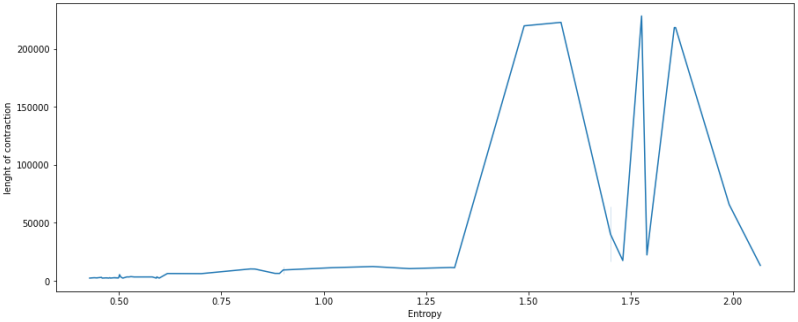
In [45]:

```
plt.figure(figsize=(15,6))
sns.scatterplot(y = df['lenght of contraction'], x = df['STD'], data = df,
             palette = 'hls')
plt.show()
```



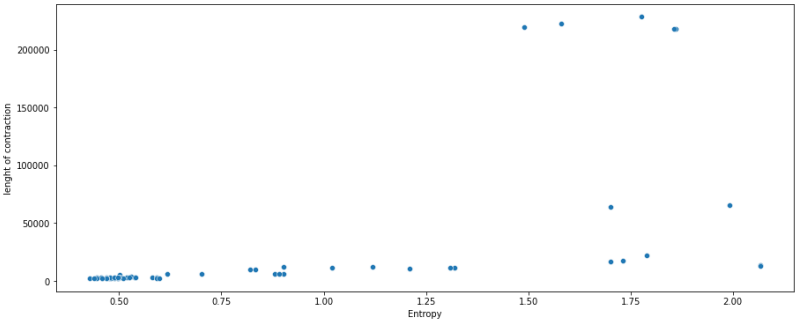
In [46]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['lenght of contraction'], x = df['Entropy'], data = df,
             palette = 'hls')
plt.show()
```



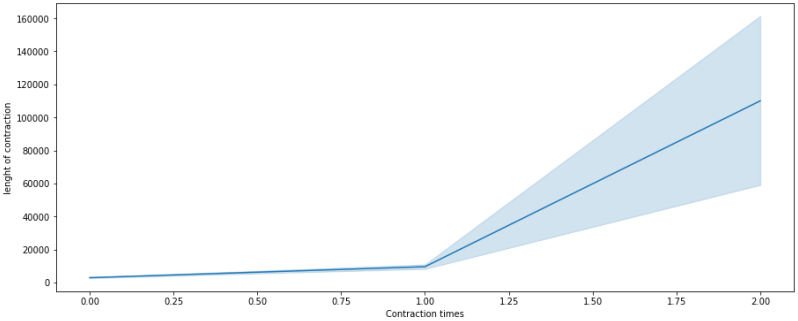
In [47]:

```
plt.figure(figsize=(15,6))
sns.scatterplot(y = df['lenght of contraction'], x = df['Entropy'], data = df,
             palette = 'hls')
plt.show()
```



In [48]:

```
plt.figure(figsize=(15,6))
sns.lineplot(y = df['lenght of contraction'], x = df['Contraction times'], data = df,
             palette = 'hls')
plt.show()
```



In [49]:

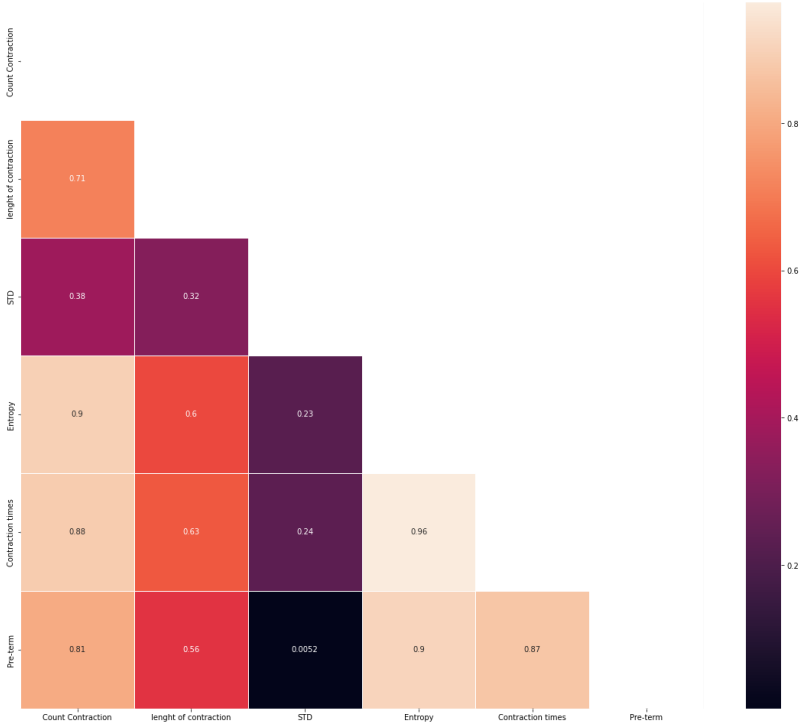
```
import numpy as np
```

In [50]:

```
df_corr = df.corr()
```

In [51]:

```
plt.figure(figsize=(20, 17))
matrix = np.triu(df_corr)
sns.heatmap(df_corr, annot=True, linewidth=.8, mask=matrix, cmap="rocket");
plt.show()
```



In [52]:

```
X = df.drop('Pre-term', axis = 1)
y = df['Pre-term']
```

In [53]:

```
from sklearn.preprocessing import StandardScaler
```

In [54]:

```
X = pd.DataFrame(StandardScaler().fit_transform(X))
```

In [55]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.20,
                                                    random_state=42)
```

In [56]:

```
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(x_train, y_train)
```

Out[56]:

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [57]:

```
y_pred= classifier.predict(x_test)
```

In [58]:

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

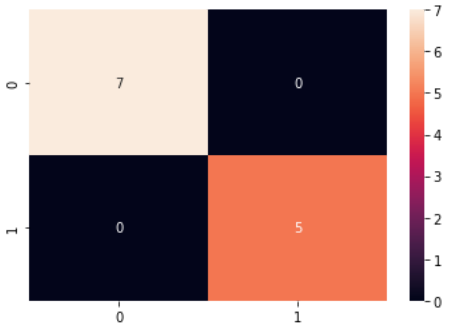
In [59]:

```
print('Confusion matrix : \n',cm)
```

```
Confusion matrix :
 [[7 0]
 [0 5]]
```

In [60]:

```
sns.heatmap(cm, annot = True)
plt.show()
```



In [61]:

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

In [62]:

```
print("\n Classification report for classifier %s:\n%s\n" % (classifier,
                                                             metrics.classification_report(y
```

```
 Classification report for classifier DecisionTreeClassifier(criterion='entrop
y', random_state=0):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       1.00      1.00      1.00         5

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12
```

In [63]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [64]:

```
rfc = RandomForestClassifier(n_estimators=10, random_state=42)
```

In [65]:

```
rfc.fit(x_train, y_train)
```

Out[65]:

```
RandomForestClassifier(n_estimators=10, random_state=42)
```

In [66]:

```
y_pred = rfc.predict(x_test)
```

In [67]:

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Accuracy: 100.00%
```

In [68]:

```
cm= confusion_matrix(y_test, y_pred)
```

In [69]:

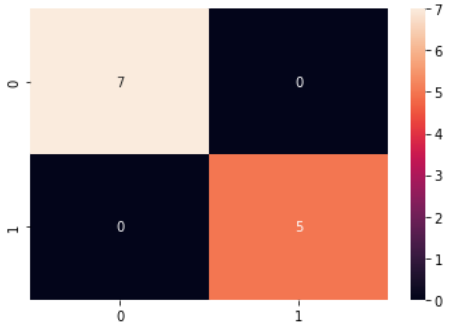
```
cm
```

Out[69]:

```
array([[7, 0],
       [0, 5]], dtype=int64)
```

In [70]:

```
sns.heatmap(cm, annot = True)
plt.show()
```



In [71]:

```
print("\n Classification report for classifier %s:\n%s\n" % (rfc,
                                                             metrics.classification_report(y
```

```
 Classification report for classifier RandomForestClassifier(n_estimators=10,
random_state=42):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       1.00      1.00      1.00         5

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12
```