EDA_downtime

October 28, 2025

0.0.1 Exploratory Data Analysis by Python

This sheet display python code for EDA for ***Manufacturing Downtime Analysis Project***. Analysis area contains:

1. ***Production performance:*** Process efficiency, sum of downtime minute, sum of scrap unit in along production period.
2. ***Downtime Factor Analysis:*** Contributing factor which cause downtime minute and scrap unit.
3. ***Downtime Occurrence Analysis:*** Trend of downtime analysis and insign correlaton of downtime occurrence and process efficiency.
4. ***Conclusion & reccommendation:*** Concluse assumption base on insigh data and propose solution to enhance process efficiency.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

```
/var/folders/f2/vt1s6tc92d5gf74rwcssrpzr0000gn/T/ipykernel_1658/3200777626.py:1:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

[2]:
```
df_production = pd.read_csv('MF_downtime.csv')
df_operators = pd.read_csv('MF_operators.csv')
df_solder = pd.read_csv('MF_solder.csv')
```

[3]:
```
df_production.columns
```

[3]:
```
Index(['Unnamed: 0', 'Run_ID', 'Date', 'Shift', 'Operator_ID', 'AC_Model',
       'Planned_Run_Time_Min', 'Actual_Production_Min', 'Downtime_Minutes',
       'Downtime_Factor', 'Actual_Output_Units', 'Target_Output_Units',
       'Scrap_Units'],
      dtype='object')
```

[4]:
```
#df_operators = df_operators.drop(columns = 'Unnamed: 0')
df_operators.rename(columns = {'Shift_Assignment': 'Shift'}, inplace = True)
df_operators.head()
```

[4]:

	Unnamed: 0	Operator_ID	Name	Years_of_Experience	Shift
0	0	OP_001	Operator 1	7	Morning
1	1	OP_002	Operator 2	4	Afternoon
2	2	OP_003	Operator 3	13	Night
3	3	OP_004	Operator 4	11	Morning
4	4	OP_005	Operator 5	8	Night

[5]:
```
df_production.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 363 entries, 0 to 362
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             363 non-null    int64
 1   Run_ID                 363 non-null    int64
 2   Date                   363 non-null    object
 3   Shift                  363 non-null    object
 4   Operator_ID            363 non-null    object
 5   AC_Model               363 non-null    object
 6   Planned_Run_Time_Min   363 non-null    int64
 7   Actual_Production_Min  363 non-null    int64
 8   Downtime_Minutes       363 non-null    int64
 9   Downtime_Factor        363 non-null    object
 10  Actual_Output_Units    363 non-null    int64
 11  Target_Output_Units    363 non-null    int64
 12  Scrap_Units            363 non-null    int64
dtypes: int64(8), object(5)
memory usage: 37.0+ KB
```

[6]:
```
df_production = df_production.drop(columns = 'Unnamed: 0')
df_production['Date'] = pd.to_datetime(df_production.Date)
df_production['Month'] = df_production.Date.dt.month
```

0.0.2 Production performance:

Summary Keys Performance Indicator

- Process efficiency: 85%

- Scraping rate(NG): 1.4%
- Average downtime minute: 31 mins/once
- Total downtime minute: 11291 mins or 188 hours
- Estimated production minute of electric board: 1.8 mins
- Model-x was leader of downtime 47%, follow by model-y 30% and model-z 21%.

Accumulated downtime minute has **buisness impact as 188 labour hours** or equal to number of ***electric board 6272 pcs***.

[7]:
```
# Create colum for process efficiency
df_production['efficiency'] = df_production.Actual_Output_Units/df_production.
 ↪Target_Output_Units * 100

# Create column for scrap rate or NG rate of production.
df_production['NG_rate'] = (df_production.Scrap_Units/df_production.
 ↪Target_Output_Units) * 100

# Create column for production minute per piece
df_production['PcsPerMinute'] = df_production.Actual_Production_Min/
 ↪df_production.Actual_Output_Units
#df_production.head()
```

[92]:
```
df_merge = pd.merge(df_production,df_operators,on = 'Operator_ID',how = 'left')
df_merge = df_merge.drop(columns = 'Unnamed: 0')
```

Summation of keys metrics

[363]:
```
df_production[['Target_Output_Units','Actual_Output_Units','Scrap_Units','Downtime_Minutes']].
 ↪sum()
```

[363]:
```
Target_Output_Units    103129
Actual_Output_Units     87531
Scrap_Units              1561
Downtime_Minutes        11291
dtype: int64
```

Average value of keys metrics

[364]:
```
df_merge[['Target_Output_Units','Scrap_Units','NG_rate','efficiency','Downtime_Minutes']].
 ↪mean()
```

[364]:
```
Target_Output_Units    284.101928
Scrap_Units              4.300275
NG_rate                  1.512749
efficiency              84.880703
Downtime_Minutes        31.104683
dtype: float64
```

0.0.3 Product Model Performance

- **47% of product volume contributed By Model-X**, 32% and 20% for model-y and z respectively. interm of downtime minute and scrap unit also according to production volume.
- **Product volume trends:** average production volume is ***up trend for all model since January to April***.
- **Downtime minutes trends:** sum of downtime minutes *end up with down trend in product-y and z (-10% and -17% change respectively)* for product x a bit increase 5%(comparison between Jan and Apr).
- **Downtime minute swing direction**: *Product-z has opposite trend direction from model-x and y.*
- **Reccommend** to investigate production process and part structure different between product model-x,y and z-pro.

[133]:
```
print(f'Part per minute of production in each product model\n\n{df_merge.
    ↪groupby('AC_Model').PcsPerMinute.mean()}')
```

```
Part per minute of production in each product model

AC_Model
Model_X        1.824459
Model_Y        1.827100
Model_Z_Pro    1.815499
Name: PcsPerMinute, dtype: float64
```

[134]:
```
print(f'Sum of value performance in production\n\n{df_merge.
    ↪groupby('AC_Model')[['Actual_Output_Units','Downtime_Minutes','Scrap_Units','NG_rate']].
    ↪sum()}')
```

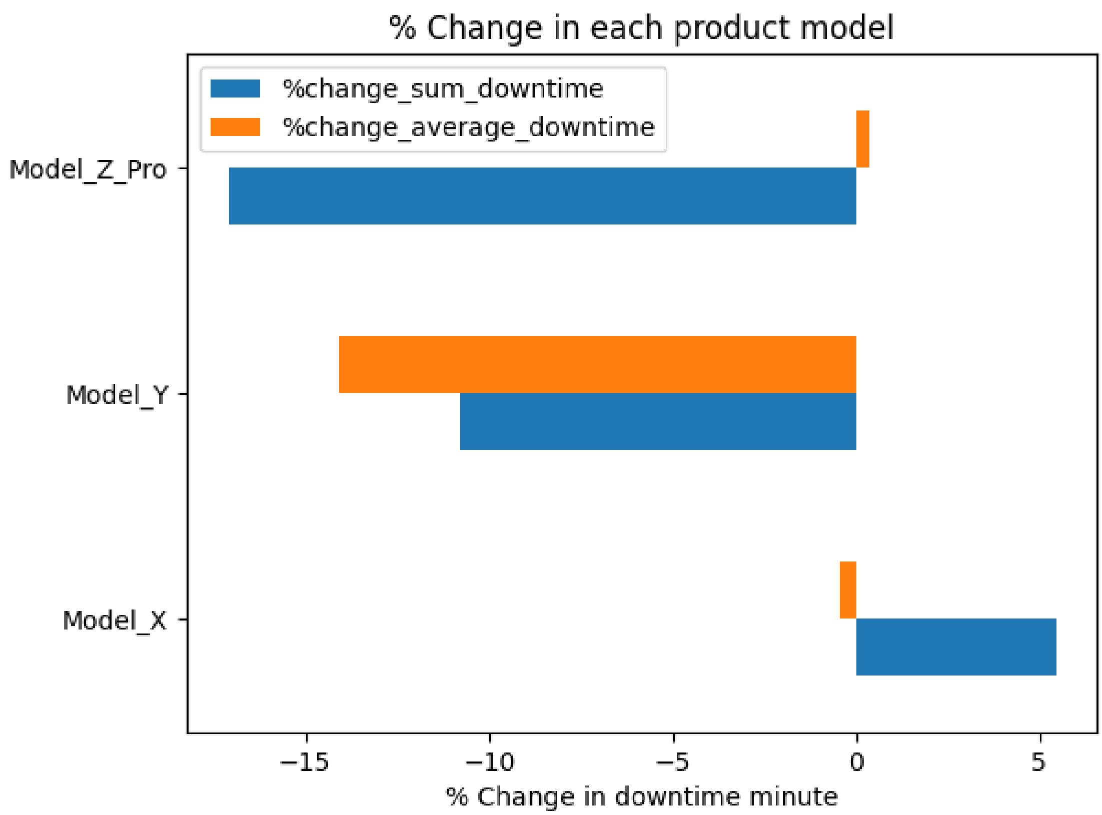
```
Sum of value performance in production

             Actual_Output_Units  Downtime_Minutes  Scrap_Units     NG_rate
AC_Model
Model_X                    41544              5307          644  226.751499
Model_Y                    28044              3601          450  158.615862
Model_Z_Pro                17943              2383          467  163.760600
```

[395]:
```
# Product downtime trends
# Determine % change in downtime minute by product model.
avg_items = {}
for i in mask.AC_Model.unique():
    month_1 = round(mask[(mask.Month == 1) & (mask.AC_Model == i)].
    ↪Downtime_Minutes.mean(),2)
    month_4 = round(mask[(mask.Month == 4) & (mask.AC_Model == i)].
    ↪Downtime_Minutes.mean(),2)
    percent = round((month_4 - month_1)/month_1 * 100,2)
    dict = {i: percent}
    avg_items.update(dict)
```

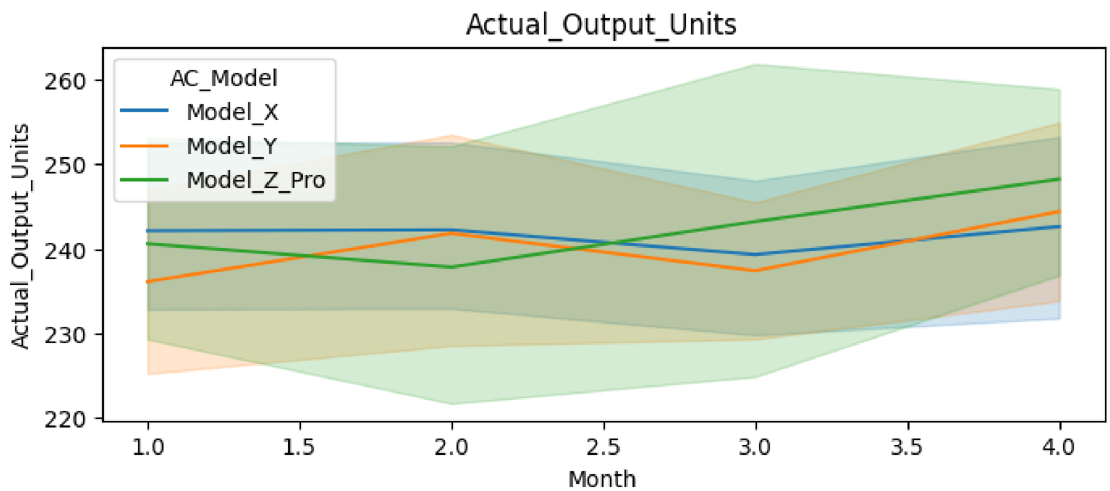
```
sum_items = {}
for i in mask.AC_Model.unique():
    month_1 = round(mask[(mask.Month == 1) & (mask.AC_Model == i)].
 ↪Downtime_Minutes.sum(),2)
    month_4 = round(mask[(mask.Month == 4) & (mask.AC_Model == i)].
 ↪Downtime_Minutes.sum(),2)
    percent = round((month_4 - month_1)/month_1 * 100,2)
    dict = {i: percent}
    sum_items.update(dict)

data = {'%change_sum_downtime': sum_items,
        '%change_average_downtime': avg_items
       }
compare_trend = pd.DataFrame(data)
print(f'% Change in each product model(since January to␣
 ↪April)\n{compare_trend}')

compare_trend.plot(kind = 'barh')
plt.title('% Change in each product model')
plt.xlabel('% Change in downtime minute')
```

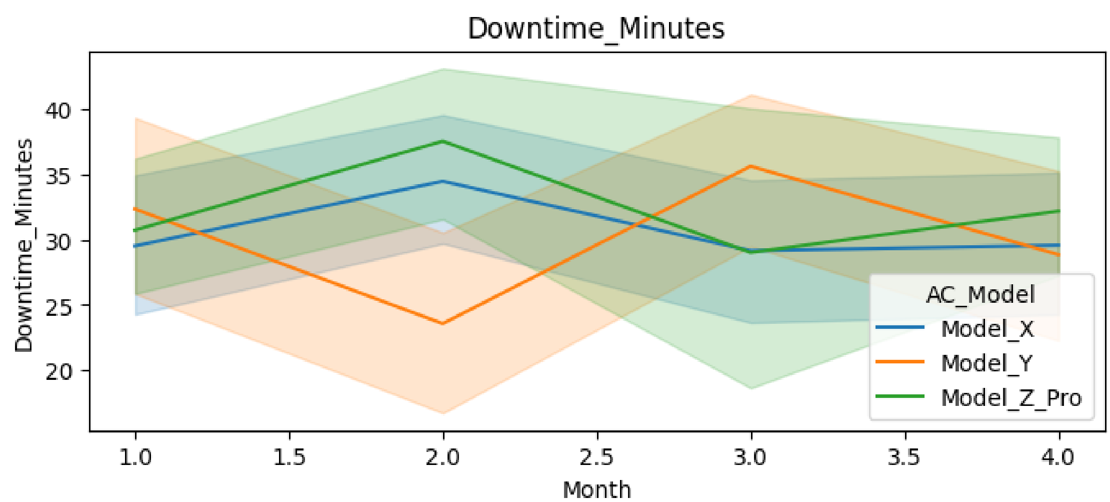
```
% Change in each product model(since January to April)
             %change_sum_downtime  %change_average_downtime
Model_X                      5.44                     -0.42
Model_Y                    -10.78                    -14.09
Model_Z_Pro                -17.10                      0.37
```

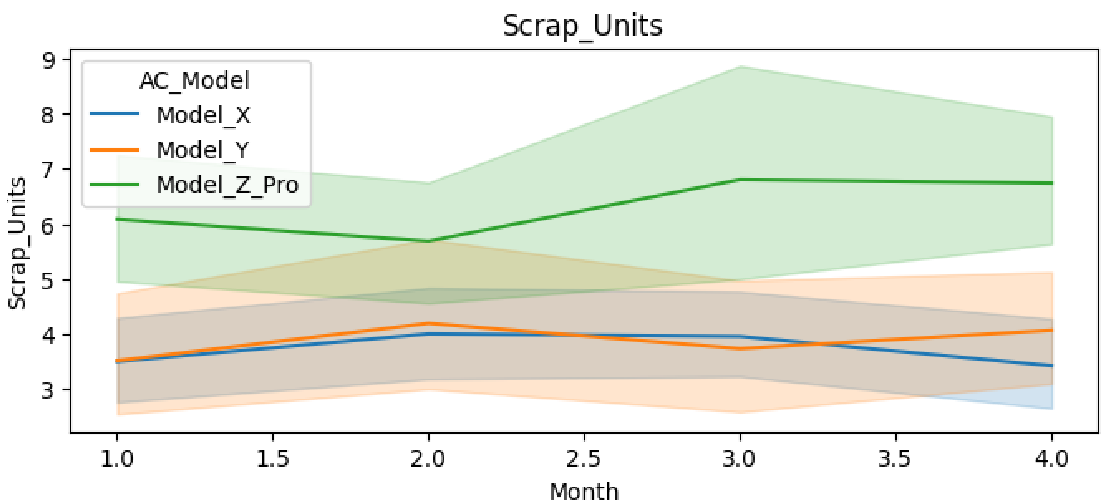
[395]:
```
Text(0.5, 0, '% Change in downtime minute')
```

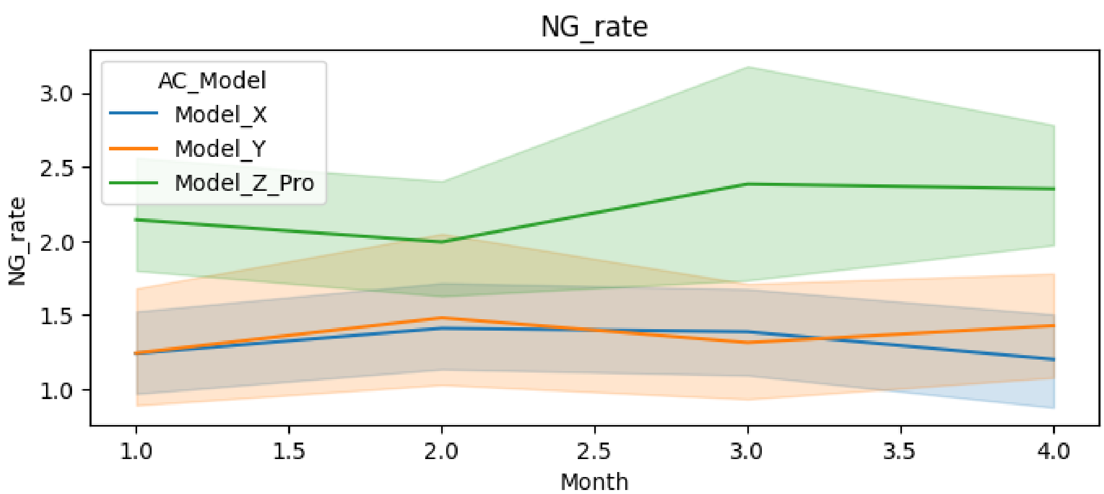


[147]:
```
for i in ['Actual_Output_Units','Downtime_Minutes','Scrap_Units','NG_rate']:
    plt.figure(figsize=(8,3))
    sns.lineplot(
        data = df_merge,
        x = df_merge.Month,
        y = df_merge[i],
        estimator = 'mean',
        hue = 'AC_Model'
    )
    plt.title(i)
    plt.show()
```


Actual_Output_Units
AC_Model
Model_X
Model_Y
Model_Z_Pro
Actual_Output_Units
260
250
240
230
220
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Month


Downtime_Minutes
AC_Model
Model_X
Model_Y
Model_Z_Pro
Downtime_Minutes
40
35
30
25
20
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Month

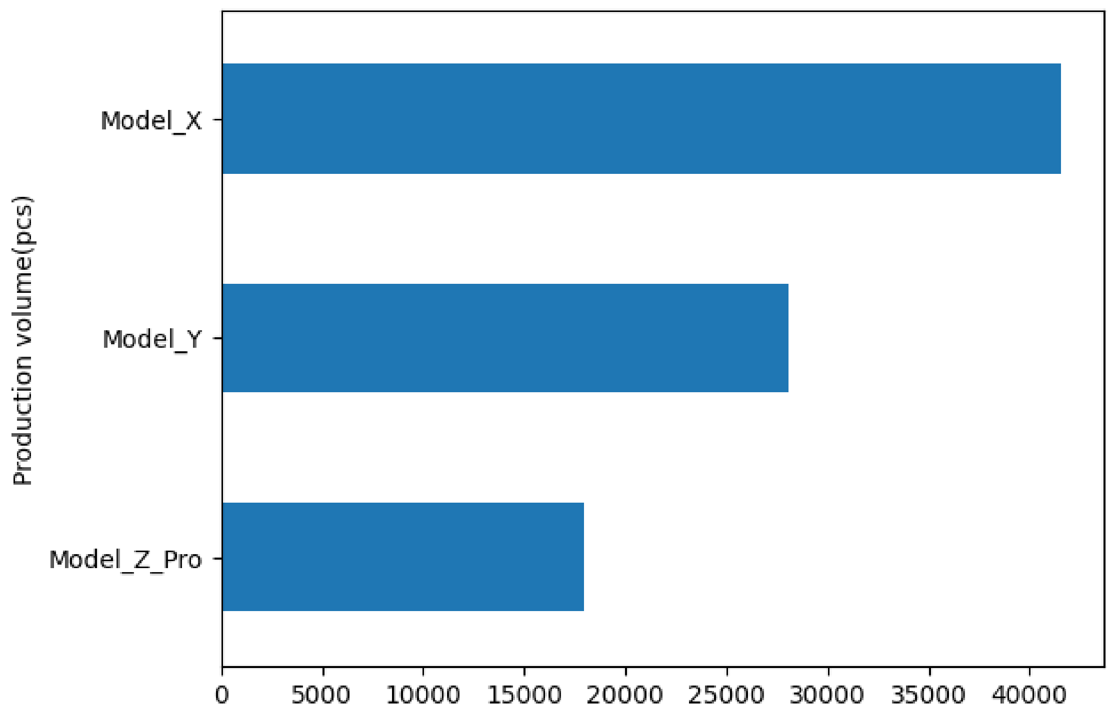




[396]:
```
pivot_model = pd.pivot_table(
    df_merge,
    index ='AC_Model',
    #columns = 'AC_Model',
    values = 'Actual_Output_Units',
    aggfunc = 'sum'
)
pivot_model['percent'] = pivot_model.Actual_Output_Units.apply(lambda x: x/
 ↪pivot_model.sum()*100)
print(f'Sum of production volume in each product mode\n{pivot_model}')
```

```
pivot_model.sort_values(by = 'Actual_Output_Units',ascending = True).
  ↪Actual_Output_Units.plot(kind = 'barh')
plt.ylabel('Production volume(pcs)')
plt.xticks(rotation = 0)
plt.show()
```

```
Sum of production volume in each product mode
             Actual_Output_Units    percent
AC_Model
Model_X                    41544  47.462042
Model_Y                    28044  32.038935
Model_Z_Pro                17943  20.499023
```



0.0.4

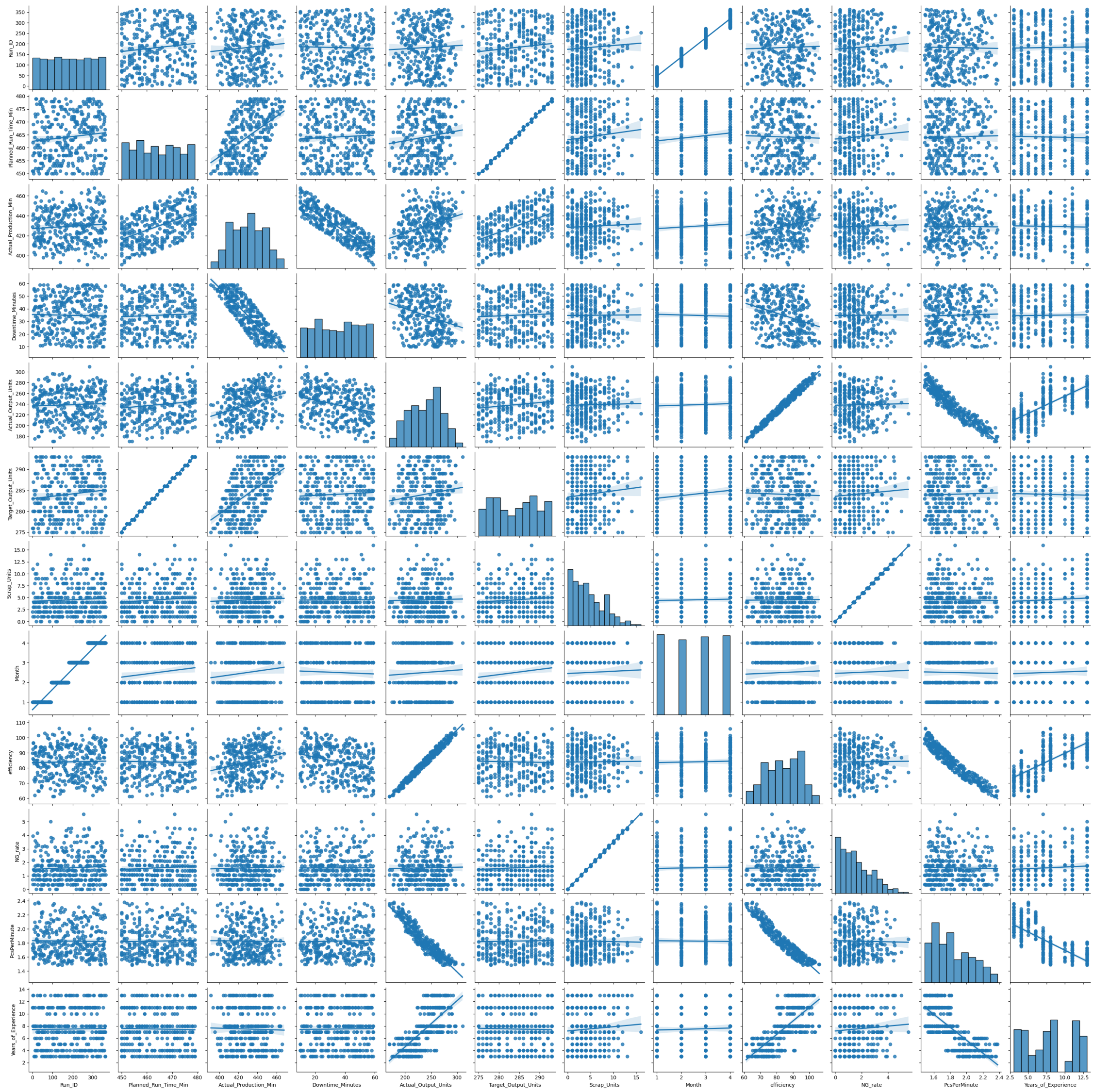
0.0.5 Correlation check

Summary

- Actual production minutes significantly reduce when actual production minute increase(coefficient -0.81).
- Currently, average production minute around 430 minutes or estimated downtime minute at 30 minutes per once.
- However, process efficiency and downtime minute has weak correlation at coefficent -0.28, so even downtime minute reduce doesn't improve process efficiency as significant.

[397]:
```python
mask = df_merge[df_merge.Downtime_Minutes != 0]
#mask.Downtime_Minutes.value_counts()
sns.pairplot(mask,kind='reg')
```

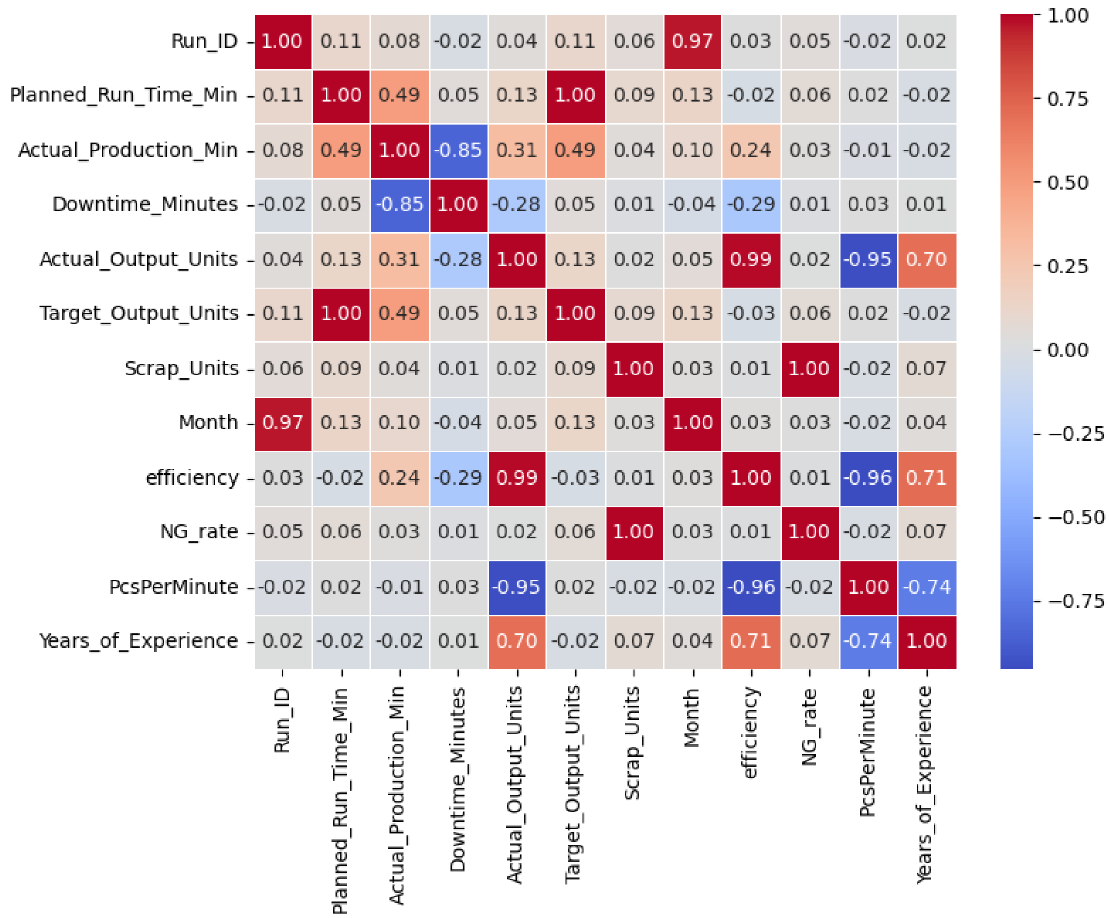
[397]: <seaborn.axisgrid.PairGrid at 0x31f855580>



[398]:
```python
orr_matrix = mask.corr(numeric_only = True)

plt.figure(figsize=(8, 6)) # Optional: Adjust the size of the plot
sns.heatmap(
    corr_matrix,
    annot=True,         # Show the correlation values on the map
    cmap='coolwarm',    # Color palette (coolwarm, viridis, magma, etc.)
```

```
    fmt=".2f",          # Format the annotation to two decimal places
    linewidths=.5,      # Lines between the cells
    cbar=True           # Show the color bar
)
```

[398]: <Axes: >



[ ]:

0.1 Downtime analysis

Downtime trend analysis

- There small change in downtime minute ***in Feb'24 average downtime increase around 6%***(30 to 32 mins), ***after Feb'24 trends to decrease to same level as Jan'24*** at end of Apr'24(29 mins)
- Average Scrap rate a bit increase during Jan - Feb(0.1%) after Feb'24 scrap unit quite constant at 1.5%.

Shift assignment analysis

- ***The lowest performance was conducted in afternoon shift***(process efficiency: 77%) and low number of operator experience year(average 5 years).
- Found *strong correlation between operator experience and process efficiency* (coeff: 0.71).
- **Assumption** operator with *high experience able to manage error in production better than operator with less experience.*
- **Reccomend:** *allocate operator with high service years to afternoon shift* to increase efficiency of process.

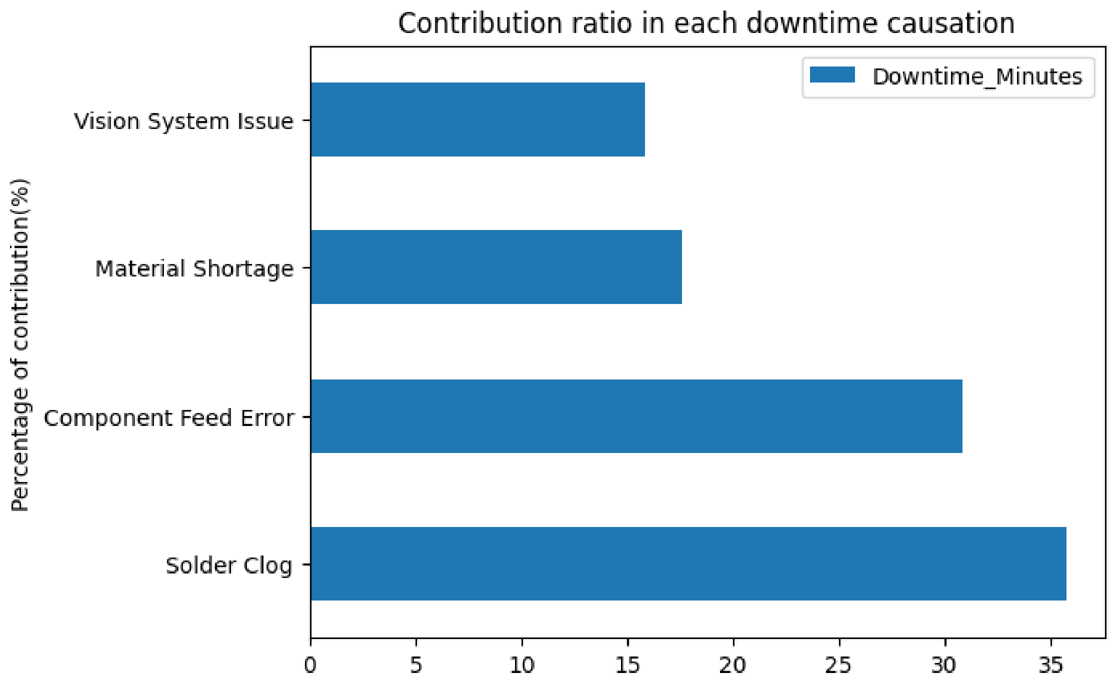
Downtime causation analysis

- **Reccomend prioritize investigate on 'Solder clog' and 'Compoenet feed error issue first** becaue of high severity level in downtime minute and scraping rate 2.9.
- **35% of downtime minute** was contributed by ***Solder clog***. Then follow by ***component feed error 30%***. Material shortage(17%) and vision system issue 15%.
- ***Compoenent feed error issue is the highest average downtime minutes(37 mins)***. Follow by Vission system(36 mins), compoent feed error(35 mins) and solder clog(33 mins). Overview, there's no much different in each causation.
- **Downtime causation trends:** sum of downtime minute is **up trends in 'Solder Clog' and 'Material Shortage'** for +16% and +7% respectively since January month while **'Vision system issue' and 'Component feed error' are opposite with down trends** since January. **Interpret trends: frequency of 'Solder clog' and 'Material Shortage' increase during period**. Confirm by **sum of downtime increase while average downtime reduce**.
- **Strong negative correlation** was found between ***downtime minute and actual production minute***(corr coeff = -0.85).Same correlation pattern as individaul downtime causation(less than - 0.80). Assumpation:

[362]:
```
# Downtime minute summary separate by downtime factor.
pivot_dt = pd.pivot_table(df_production,
               index ='Downtime_Factor',
               values = 'Downtime_Minutes',
               aggfunc = 'sum'
              )

pivot_dt = pivot_dt.apply(lambda x: x/pivot_dt.Downtime_Minutes.sum()*100).
  ↪sort_values(by = 'Downtime_Minutes',ascending= False)
pivot_filter = pivot_dt[pivot_dt.Downtime_Minutes>0]
print(f'Contribution ratio of downtime causation(%)\n{pivot_filter}')

pivot_filter.plot(kind = 'barh')
plt.ylabel('Percentage of contribution(%)')
plt.xticks(rotation = 0)
plt.title('Contribution ratio in each downtime causation')
plt.show()
```

```
Contribution ratio of downtime causation(%)
                      Downtime_Minutes
```

```
Downtime_Factor
Solder Clog                35.754140
Component Feed Error       30.838721
Material Shortage          17.544947
Vision System Issue        15.862191
```

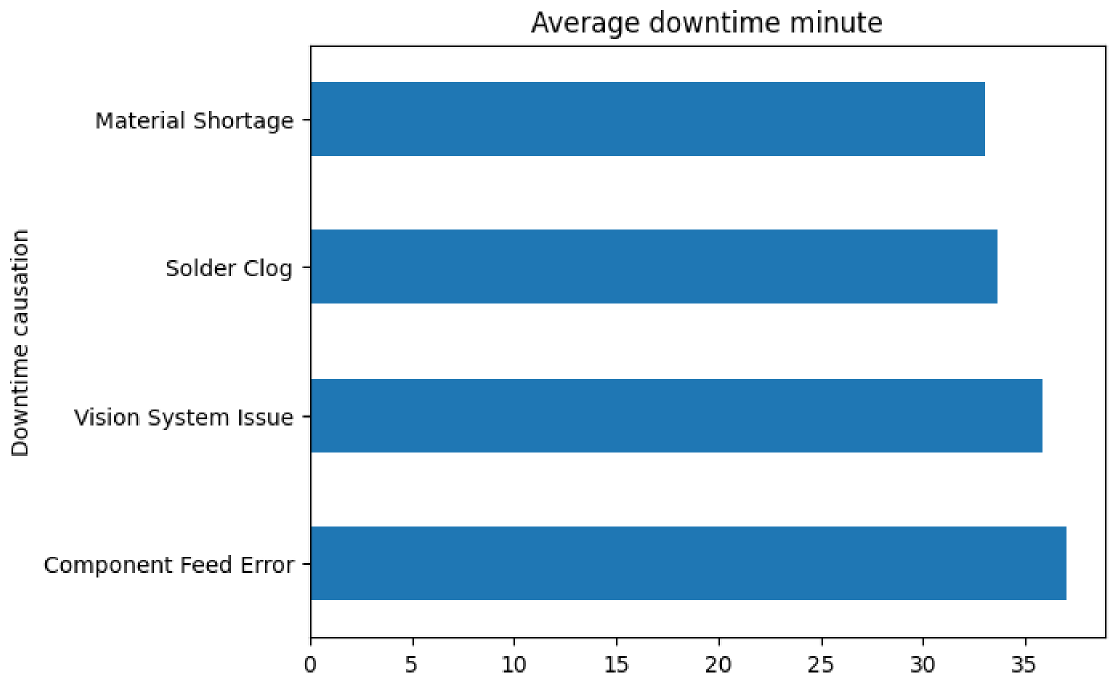


[162]:
```python
# Determine average minute in each downtime factor.
downtime_avg = df_production.groupby('Downtime_Factor').Downtime_Minutes.mean().
 ↪sort_values(ascending = False)
print(f'Average downtime minute by downtime causation\n{downtime_avg}')
```

```
Average downtime minute by downtime causation
Downtime_Factor
Component Feed Error    37.042553
Vision System Issue     35.820000
Solder Clog             33.641667
Material Shortage       33.016667
No Downtime              0.000000
Name: Downtime_Minutes, dtype: float64
```

[158]:
```python
mask.groupby('Downtime_Factor').Downtime_Minutes.mean().sort_values(ascending =␣
 ↪False).plot(kind = 'barh')
plt.xticks(rotation = 0)
plt.title('Average downtime minute')
plt.ylabel('Downtime causation')
```

```
plt.show()
```



[371]:
```
avg_items = {}
for i in mask.Downtime_Factor.unique():
    month_1 = round(mask[(mask.Month == 1) & (mask.Downtime_Factor == i)].
 ↪Downtime_Minutes.mean(),2)
    month_4 = round(mask[(mask.Month == 4) & (mask.Downtime_Factor == i)].
 ↪Downtime_Minutes.mean(),2)
    percent = round((month_4 - month_1)/month_1 * 100,2)
    dict = {i: percent}
    avg_items.update(dict)
    #print(f'Percent change(%) in average downtime minute [{i}]:␣
 ↪{percent}%\nMonth_1: {month_1}\nMonth_4: {month_4}\n')
```

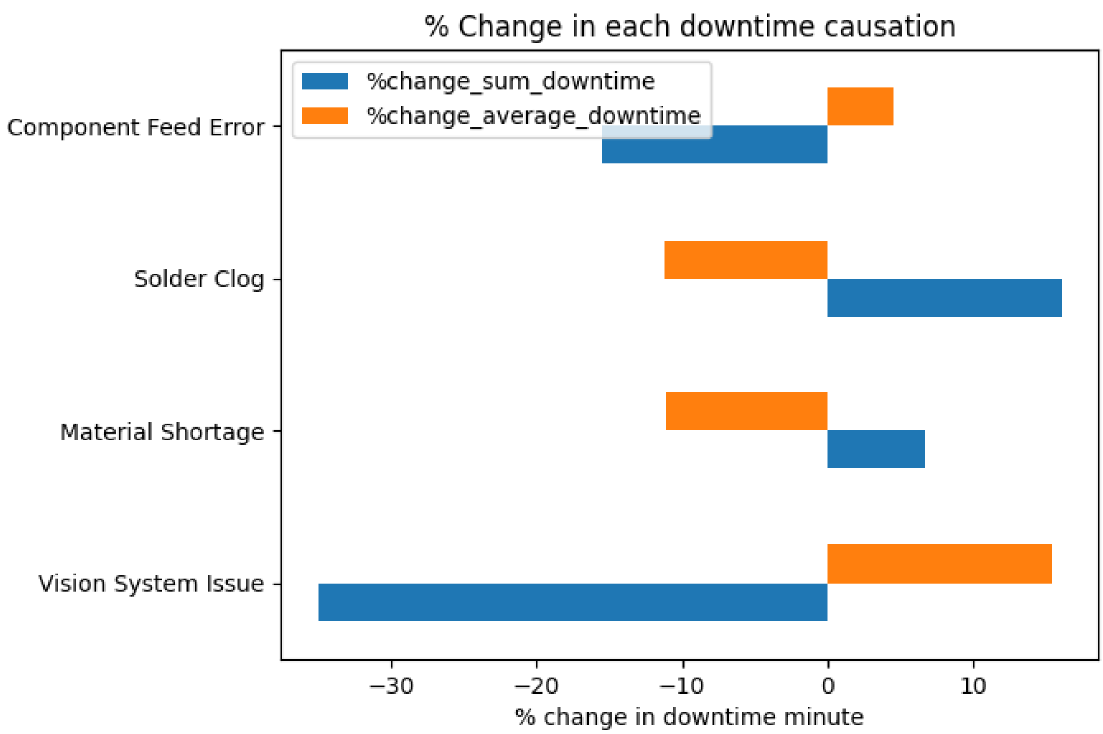
[369]:
```
sum_items = {}
for i in mask.Downtime_Factor.unique():
    month_1 = round(mask[(mask.Month == 1) & (mask.Downtime_Factor == i)].
 ↪Downtime_Minutes.sum(),2)
    month_4 = round(mask[(mask.Month == 4) & (mask.Downtime_Factor == i)].
 ↪Downtime_Minutes.sum(),2)
    percent = round((month_4 - month_1)/month_1 * 100,2)
    dict = {i: percent}
    sum_items.update(dict)
```

```
    #print(f'Percent change(%) in sum of downtime minute [{i}]:␣
↪{percent}%\nMonth_1: {month_1}\nMonth_4: {month_4}\n')
```

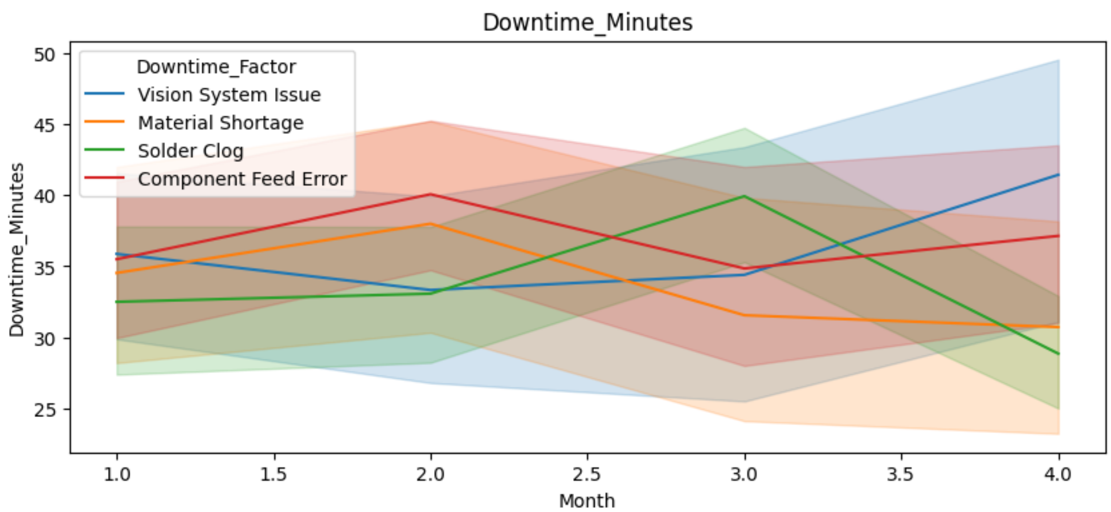
[375]:
```
data = {'%change_sum_downtime': sum_items,
        '%change_average_downtime': avg_items
       }
compare_trend = pd.DataFrame(data)
print(f'% Change in each downtime cuasation(since January to␣
↪April)\n{compare_trend}')

compare_trend.plot(kind = 'barh')
plt.title('% Change in each downtime causation')
plt.xlabel('% change in downtime minute')
```

```
% Change in each downtime cuasation(since January to April)
                      %change_sum_downtime  %change_average_downtime
Vision System Issue                 -35.02                     15.50
Material Shortage                     6.76                    -11.03
Solder Clog                          16.09                    -11.23
Component Feed Error                -15.49                      4.62
```

[375]: Text(0.5, 0, '% change in downtime minute')



[361]:
```
cols = ['Downtime_Minutes']

for col in cols:
    plt.figure(figsize=(10,4))
    sns.lineplot(
        x = mask.Month,
        y = mask[col],
        data = mask,
        hue = 'Downtime_Factor',
    estimator = 'mean')

    plt.title(col)
    plt.xticks(rotation = 0)
    plt.show()
```

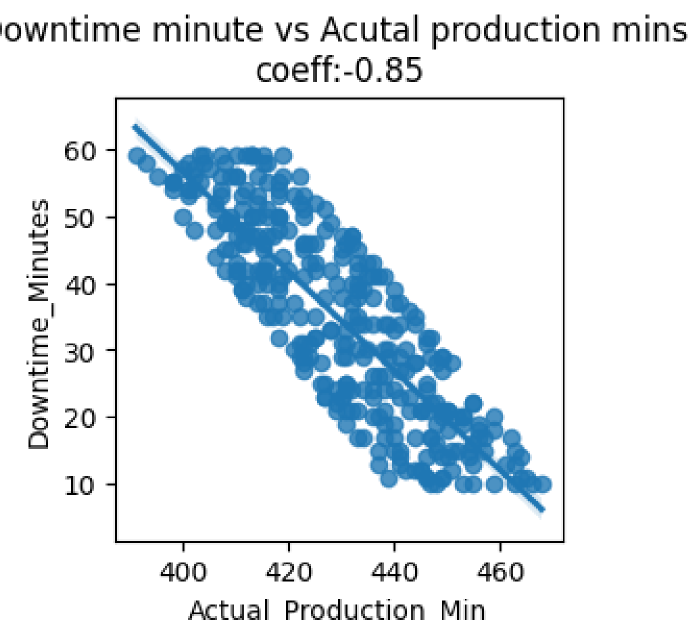


0.1.1 Deep dive into individual factor

- **Strong negative correlation** was found between ***downtime minute and actual production minute***(corr coeff = -0.85).
- **Same correlation pattern as individaul downtime causation**(less than - 0.80).

[530]:
```
corr_matrix = mask.corr(numeric_only = True)
coeff = corr_matrix['Actual_Production_Min'].loc['Downtime_Minutes']

# Regression plot
x = mask.Actual_Production_Min
y = mask.Downtime_Minutes
plt.figure(figsize=(3,3)) # Optional: Sets the figure size

#plt.scatter(x, y, color='darkred', marker='o', s=100, alpha=0.7)
```

```
sns.regplot(x = x, y= y, data = mask)
plt.title('Downtime minute vs Acutal production mins' + ' ' +  '\n' + 'coeff:'␣
 ↪+ str(round(coeff,2)))
plt.show()

print(f'Current average production minute: {round(df_merge.
 ↪Actual_Production_Min.mean(),2)} minutes')
print(f'Current average downtime minute: {round(df_merge.Downtime_Minutes.
 ↪mean(),2)} minutes')
```




```
Current average production minute: 433.15 minutes
Current average downtime minute: 31.1 minutes
```

[541]:
```
df_merge.head()
```

[541]:
```
   Run_ID        Date    Shift_x Operator_ID     AC_Model  \
0       1 2024-01-01    Morning      OP_004      Model_X
1       2 2024-01-01  Afternoon      OP_006      Model_Y
2       3 2024-01-01      Night      OP_018      Model_X
3       4 2024-01-02    Morning      OP_004  Model_Z_Pro
4       5 2024-01-02  Afternoon      OP_006  Model_Z_Pro

   Planned_Run_Time_Min  Actual_Production_Min  Downtime_Minutes  \
0                   451                    429                22
1                   462                    436                26
2                   457                    440                17
```

```
3                   451                    421                 30
4                   469                    442                 27

       Downtime_Factor  Actual_Output_Units  Target_Output_Units  Scrap_Units  \
0  Vision System Issue                  248                  276            2
1    Material Shortage                  244                  283            1
2          Solder Clog                  259                  280           11
3    Material Shortage                  266                  276            6
4  Vision System Issue                  277                  287            3

   Month  efficiency   NG_rate  PcsPerMinute        Name  \
0      1   89.855072  0.724638      1.729839  Operator 4
1      1   86.219081  0.353357      1.786885  Operator 6
2      1   92.500000  3.928571      1.698842 Operator 18
3      1   96.376812  2.173913      1.582707  Operator 4
4      1   96.515679  1.045296      1.595668  Operator 6

   Years_of_Experience    Shift_y
0                   11    Morning
1                   13  Afternoon
2                    8      Night
3                   11    Morning
4                   13  Afternoon
```

[518]:

```python
# Correlation coefficient in each downtime causation
print('correleation coefficient separate by <downtime causation>')
for i in mask.Downtime_Factor.unique():
    mask2 = mask[mask.Downtime_Factor == i]
    #Correlation coefficient
    corr_matrix = mask2.corr(numeric_only = True)
    corr_mask = corr_matrix[(corr_matrix.Downtime_Minutes > 0.8) | (corr_matrix.
 ↪Downtime_Minutes < -0.8)].Downtime_Minutes
    coeff = corr_mask.loc['Actual_Production_Min']
    print(f'[{i}]: {round(coeff,2)}')
```

```
correleation coefficient separate by <downtime causation>
[Vision System Issue]: -0.82
[Material Shortage]: -0.85
[Solder Clog]: -0.86
[Component Feed Error]: -0.84
```

[519]:

```python
print('correleation coefficient separate by <shift assignment>')
for i in mask.Shift_x.unique():
    mask2 = mask[mask.Shift_x == i]
    #Correlation coefficient
    corr_matrix = mask2.corr(numeric_only = True)
    coeff = corr_matrix.Downtime_Minutes.loc['Actual_Production_Min']
```

```
    print(f'[{i}]: {coeff}')

    #plt.figure(figsize=(3,2))
    #sns.regplot(data = mask,
    #x = 'Actual_Production_Min',
    #y= 'Downtime_Minutes')
    #plt.title('Shift: ' + i + '\n'+'coeff:' + str(round(coeff,2)))
    #plt.show()
```

```
correleation coefficient separate by <shift assignment>
[Morning]: -0.8340025241449053
[Afternoon]: -0.8572992524421759
[Night]: -0.8442503937667535
```

[510]:
```
print('correleation coefficient separate by <Years of Experience>')
for i in mask.Years_of_Experience.unique():
    mask2 = mask[mask.Years_of_Experience == i]
    #Correlation coefficient
    corr_matrix = mask2.corr(numeric_only = True)
    coeff = corr_matrix.Downtime_Minutes.loc['Actual_Production_Min']
    print(f'[{i}]: {coeff}')
```

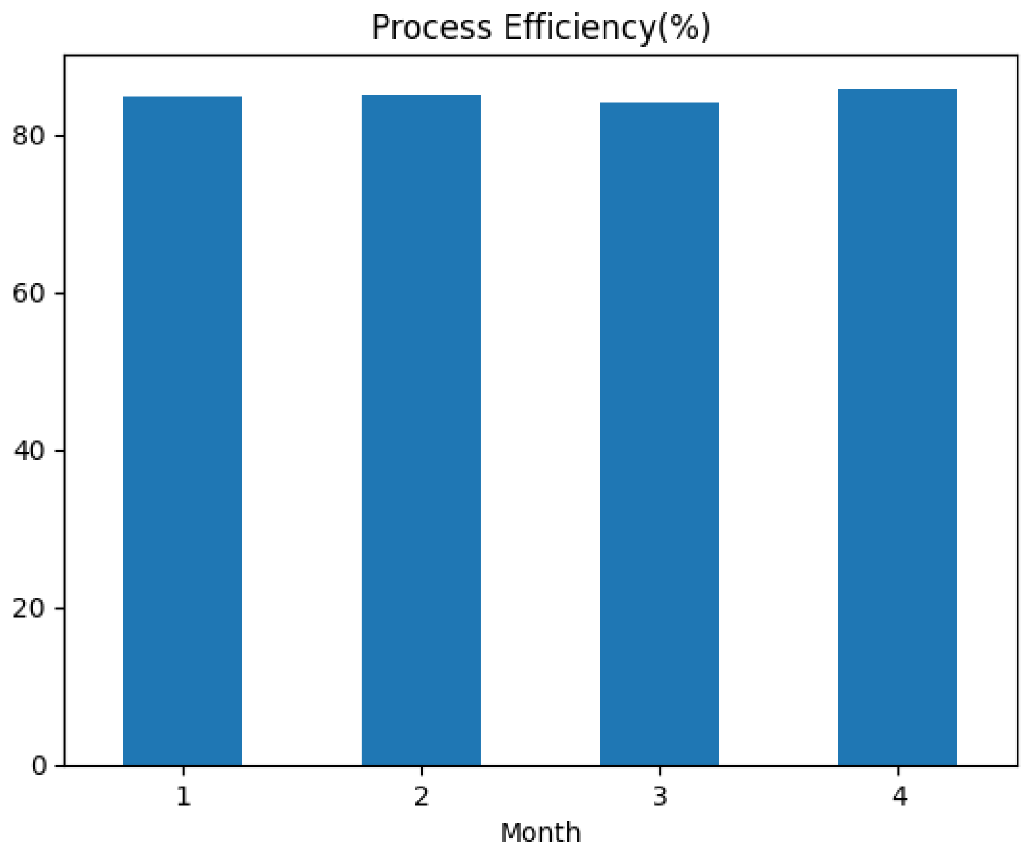
```
correleation coefficient separate by <Years of Experience>
[11]: -0.8407859706298585
[13]: -0.8301566986179292
[8]: -0.872962829517277
[3]: -0.8709099348401472
[6]: -0.7730516998216674
[4]: -0.8468921808072551
[7]: -0.8141790655335622
[5]: -0.8256582878263454
[10]: -0.9232371535870427
```

[407]:
```
# Monthly analysis
df_month = df_production.groupby('Month')[['Actual_Production_Min',␣
 ↪'Downtime_Minutes','Scrap_Units','NG_rate','efficiency']].mean()
display(df_month)
df_month.efficiency.plot(kind = 'bar')
plt.title('Process Efficiency(%)')
plt.xticks(rotation = 0)
```

```
       Actual_Production_Min  Downtime_Minutes  Scrap_Units   NG_rate  \
Month
1                 431.763441         30.752688     4.172043  1.474200
2                 431.218391         32.379310     4.356322  1.534951
3                 434.161290         31.473118     4.333333  1.521361
4                 435.411111         29.855556     4.344444  1.522223

       efficiency
```

```
Month
1        84.705490
2        85.059474
3        83.970537
4        85.829451
```

[407]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, '1'), Text(1, 0, '2'), Text(2, 0, '3'), Text(3, 0, '4')])
```

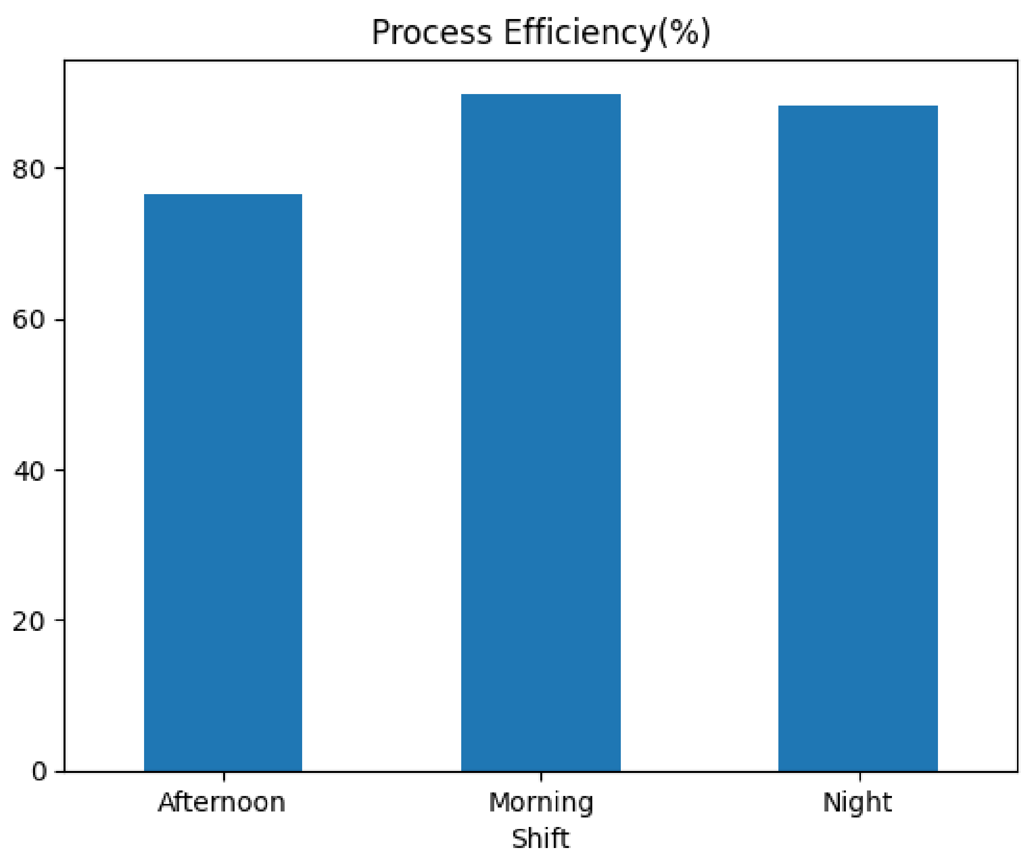


[23]:
```
# Daily analysis
pivot = pd.pivot_table(
    df_production,
    index = 'Shift',
    values =␣
 ↪['Downtime_Minutes','efficiency','NG_rate','Scrap_Units','Actual_Output_Units','Target_Outp
    #columns = 'Shift_x',
    aggfunc = 'mean'
)
display(pivot)
pivot.efficiency.plot(kind = 'bar')
```

```
plt.title('Process Efficiency(%)')
plt.xticks(rotation = 0)
```

```
           Actual_Output_Units  Downtime_Minutes   NG_rate  Scrap_Units  \
Shift
Afternoon           217.438017         32.677686  1.374427     3.909091
Morning             255.140496         27.561983  1.550984     4.404959
Night               250.818182         33.074380  1.612836     4.586777

           Target_Output_Units  efficiency
Shift
Afternoon           284.206612   76.514602
Morning             284.132231   89.812676
Night               283.966942   88.314831
```

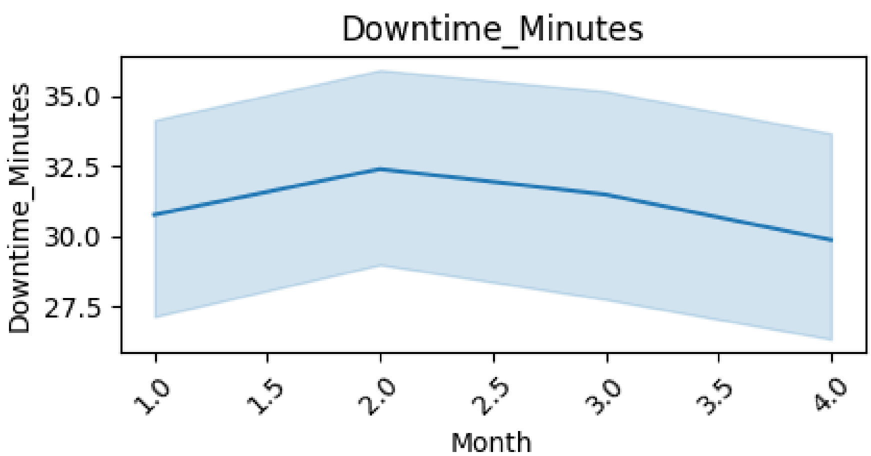
[23]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'Afternoon'), Text(1, 0, 'Morning'), Text(2, 0, 'Night')])
```

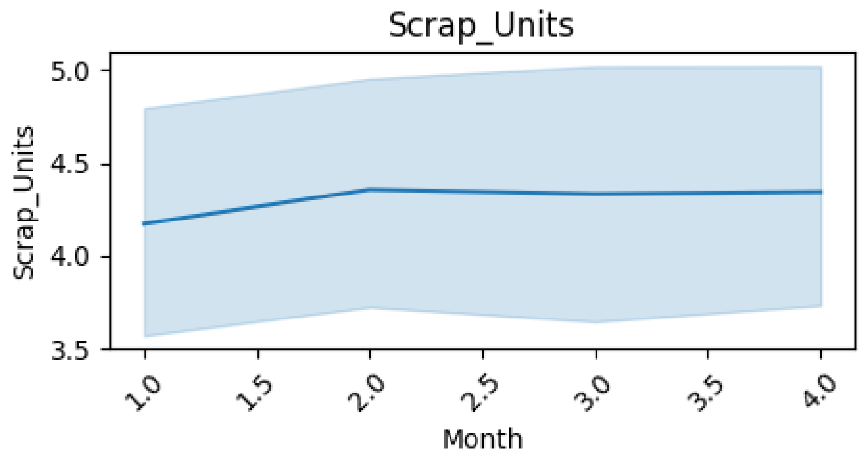


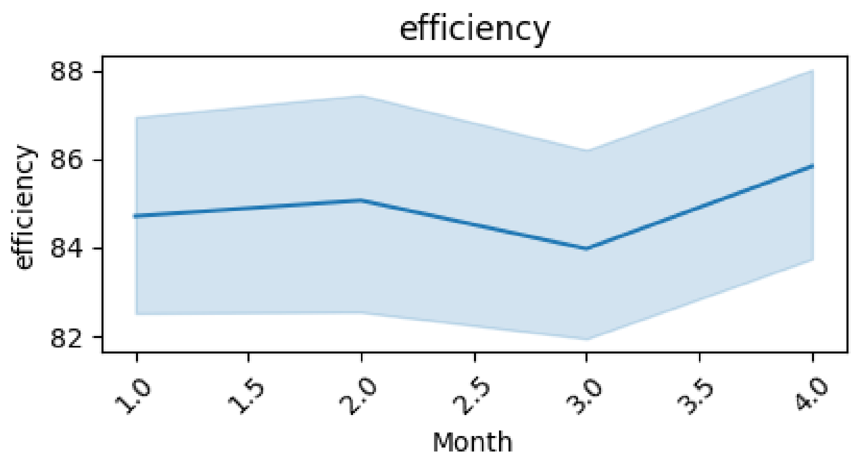
[24]:
```
df_production.columns
```

[24]: Index(['Run_ID', 'Date', 'Shift', 'Operator_ID', 'AC_Model',
       'Planned_Run_Time_Min', 'Actual_Production_Min', 'Downtime_Minutes',
       'Downtime_Factor', 'Actual_Output_Units', 'Target_Output_Units',
       'Scrap_Units', 'Month', 'efficiency', 'NG_rate', 'PcsPerMinute'],
      dtype='object')

[426]:
```python
cols = ['Downtime_Minutes','Scrap_Units','efficiency']
for col in cols:
    plt.figure(figsize=(5,2))
    sns.lineplot(
        x = df_production.Month,
        y = df_production[col],
        data = df_production)
        #hue = 'Shift' )

    plt.title(col)
    plt.xticks(rotation = 45)
    plt.show()
```





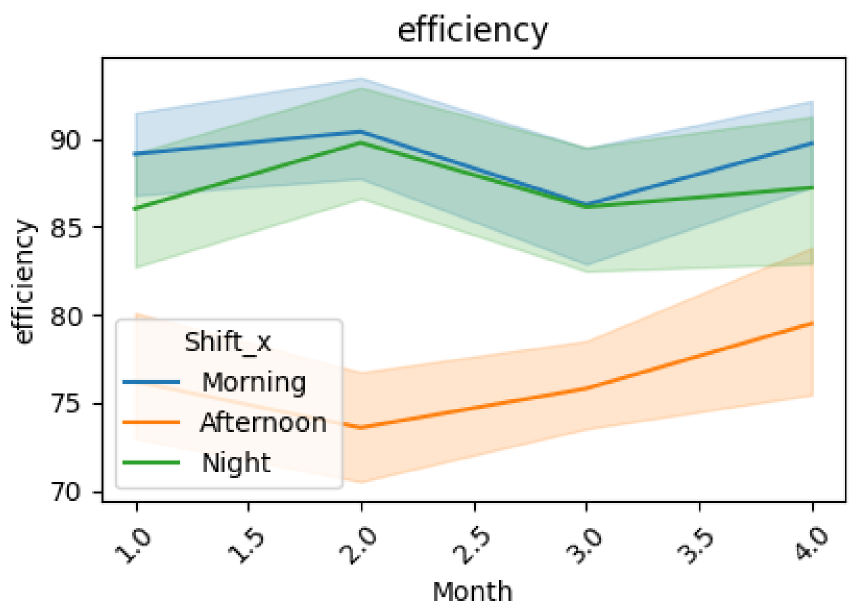


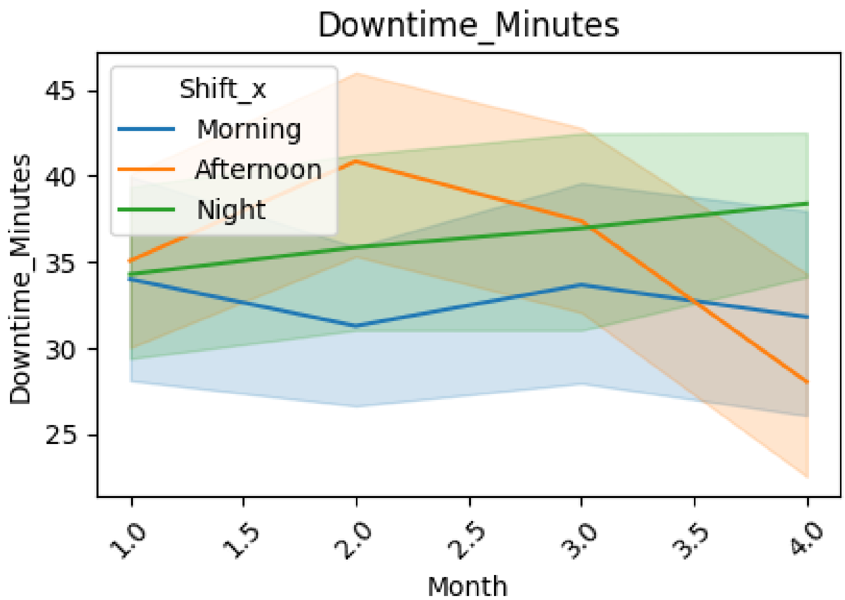
Shift assignment analysis(Morning/Afternoon/Night)

- **Average Downtime Trends:** In January, afternoon and night shift trend up but morning go down. After Febuary afternoon and morning shift is down trend but night shift go up until end of April.
- **Efficiency trends:** Morning and night shift are similar swing movement in period back to same level as January at end of April. For Afternoon shift perform with lower level than both of morning and night shift until end of April. A bit go down in Febuary after that was up trends until end of April.
- Reccomend to check suspect factor like operator, process change between morning, afternoon and night shift.

[440]:
```
cols = ['efficiency','Downtime_Minutes']
for col in cols:
```

```
plt.figure(figsize=(5,3))
sns.lineplot(
    x = df_production.Month,
    y = df_production[col],
    data = mask,
    hue = 'Shift_x',
    estimator = 'mean')

plt.title(col)
plt.xticks(rotation = 45)
plt.show()
```





summary by causation

[28]:
```
df_operators.columns
```

[28]:
```
Index(['Unnamed: 0', 'Operator_ID', 'Name', 'Years_of_Experience', 'Shift'],
dtype='object')
```

[29]:
```
df_production.shape
```

[29]:
```
(363, 16)
```

[30]:
```
df_merge = pd.merge(df_production,df_operators,on = 'Operator_ID',how = 'left')
#df_merge.info()
```

[517]:
```
pivot = pd.pivot_table(
    df_merge,
    index = 'Shift_x',
    values = ['Years_of_Experience','Downtime_Minutes','NG_rate','efficiency'],
    #columns = 'Years_of_Experience',
    aggfunc = 'mean'
)
round(pivot,2)
```

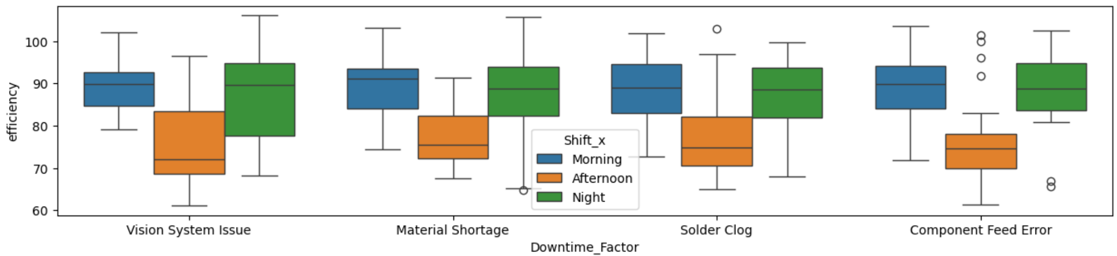
[517]:

	Downtime_Minutes	NG_rate	Years_of_Experience	efficiency
Shift_x				
Afternoon	32.68	1.37	4.90	76.51
Morning	27.56	1.55	8.22	89.81

```
Night                       33.07      1.61                    9.43        88.31
```

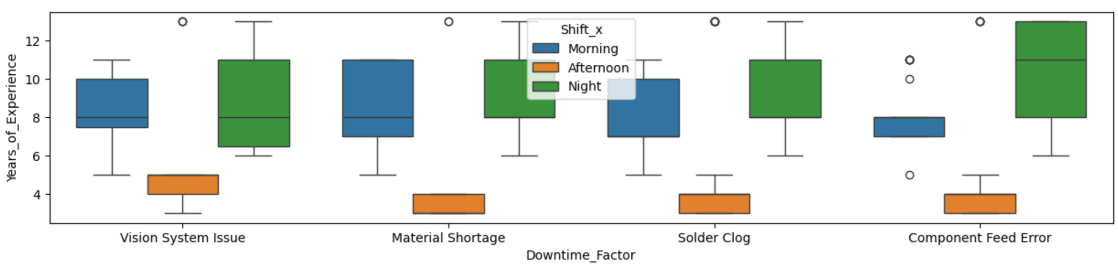
[446]:
```python
mask = df_merge[df_merge.Downtime_Minutes != 0]
plt.figure(figsize=(15,3))
sns.boxplot(
    data = mask,
    x = 'Downtime_Factor',
    y = 'efficiency',
    hue = 'Shift_x'
)
```

[446]: `<Axes: xlabel='Downtime_Factor', ylabel='efficiency'>`



[459]:
```python
mask = df_merge[df_merge.Downtime_Minutes != 0]
plt.figure(figsize=(15,3))
sns.boxplot(
    data = mask,
    x = 'Downtime_Factor',
    y = 'Years_of_Experience',
    hue = 'Shift_x'
)
```

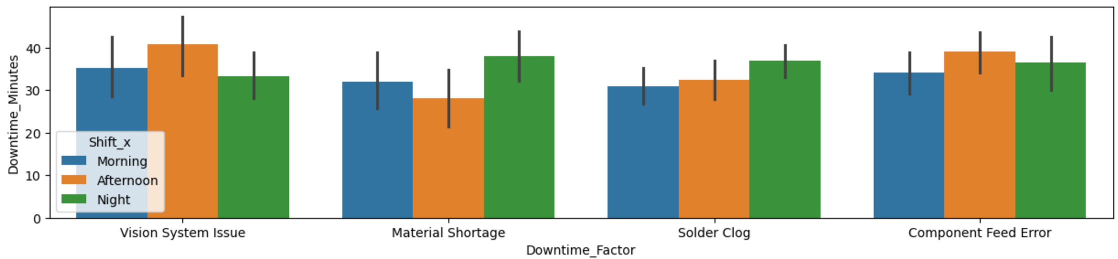
[459]: `<Axes: xlabel='Downtime_Factor', ylabel='Years_of_Experience'>`



[457]:
```python
mask = df_merge[df_merge.Downtime_Minutes != 0]
plt.figure(figsize=(15,3))
```

```
sns.barplot(
    data = mask,
    x = 'Downtime_Factor',
    y = 'Downtime_Minutes',
    hue = 'Shift_x',
    estimator = 'mean'
)
```

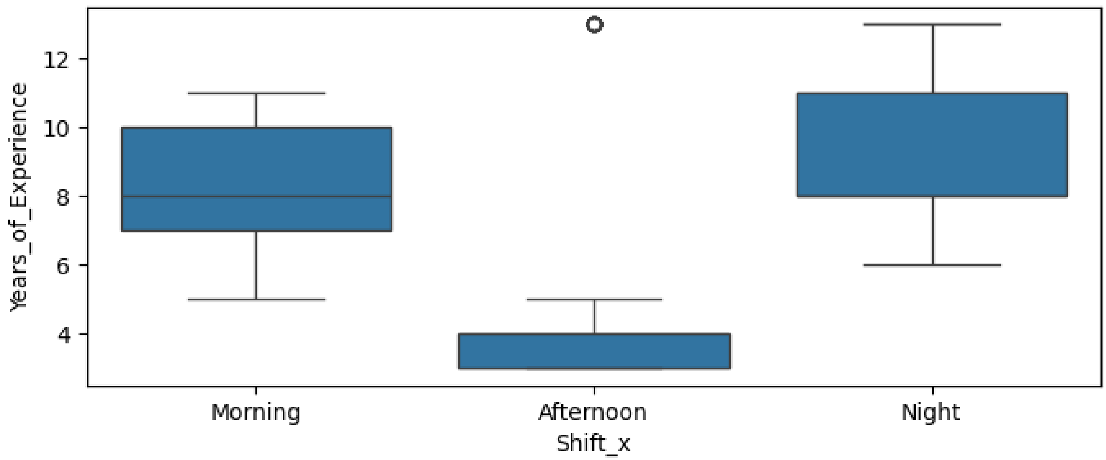
[457]: <Axes: xlabel='Downtime_Factor', ylabel='Downtime_Minutes'>



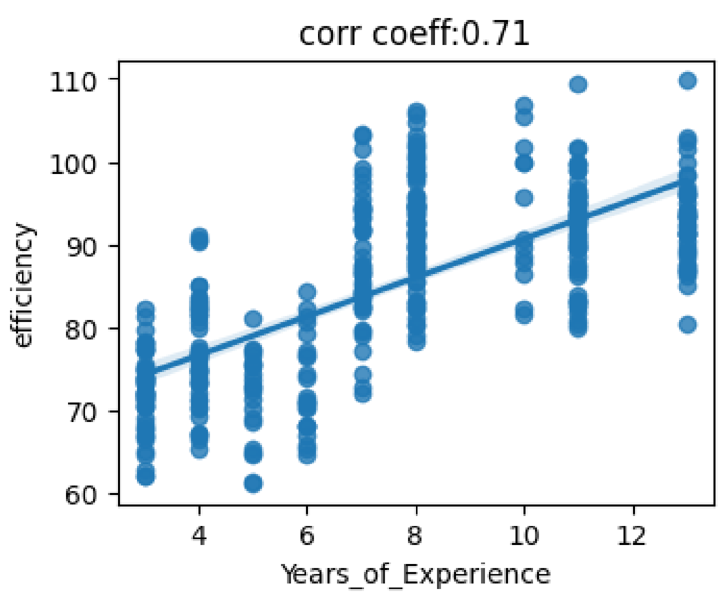
[514]:
```
shift_years = df_merge.groupby('Shift_x').Years_of_Experience.mean()
print(f'Average service year of operator in each shift␣
 ↪assignment\n{shift_years}')

plt.figure(figsize=(8,3))
sns.boxplot(
    data = mask,
    x = 'Shift_x',
    y = 'Years_of_Experience')
```

```
Average service year of operator in each shift assignment
Shift_x
Afternoon    4.900826
Morning      8.223140
Night        9.429752
Name: Years_of_Experience, dtype: float64
```

[514]: <Axes: xlabel='Shift_x', ylabel='Years_of_Experience'>



[35]:
```
#Correlation coefficient
mask = df_merge[df_merge.Downtime_Minutes != 0]
corr_matrix = mask.corr(numeric_only = True)
coeff = corr_matrix.loc['Years_of_Experience','efficiency']

# Regression plot
x = df_merge.Years_of_Experience
y = df_merge.efficiency

plt.figure(figsize=(4,3)) # Optional: Sets the figure size
sns.regplot(x = x, y= y, data = df_merge)
plt.title('corr coeff:' + str(round(coeff,2)))
plt.show()
```



- ***All low performance was operated in afternoon shift*** and low number of operator experience year.
- *There's significant positive correleation*

  between experience year and process efficiency ***(coefficient : 0.68)***
- ***Assumption: operator who has less experience year was lack of management*** in process error and factilty than high experience operator.

0.1.2 Insign information discovered

1. ***Afternoon shift is influentail that impact to prouduction efficiency and downtime factor.***
2. Then deep analysze and found that less experience year of operator influent to efficient. Assumption is high experience operator able manage error situation as better than low experience person.
3. Therefore, *focusing on high contributing causation 'Solder Clog' and 'Component Feed Error' issue first.*

0.1.3 Conclusion & Reccomendation

1. ***Quick action approach:*** Allocate high exeperience to afternoon shift to improve process efficiency.
2. ***Mid term plan of root cause analysis:*** Deep investigate root cause of downtime minute, focus high contributing factor(Solder Clog & Component Feed Error) by ask support from facility team to analyze data record of machine.