# Operations management

**1- Load libraries**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

**2- Load ops data**

df = pd.read\_csv('data/operations/data.csv')

df.head()

**3- Graph metrics**

# Plot KDE distributions for each metric

plt.figure(figsize=(15, 18))

for i, column in enumerate(df.columns[2:]): # Skip 'Well' and 'Rig'

plt.subplot(5, 2, i+1)

sns.histplot(df[column], kde=True, bins=30)

plt.title(f'{column} Distribution')

plt.tight\_layout()

plt.show()

**4- Compute KPIs**

from scipy.stats import norm

# Generate the equivalent normal distribution for each KPI

for col in df.columns[2:]:

mu, std = norm.fit(df[col])

df[f'{col}\_norm'] = norm(mu, std).pdf(df[col])

# Calculate the 90% confidence interval for each KPI

ci = norm.interval(0.9, loc=mu, scale=std)

df[f'{col}\_ci'] = ci[1] - ci[0]

# Plot the distribution and confidence interval for each KPI

fig, ax = plt.subplots()

sns.histplot(df, x=col, kde=True, stat='density', ax=ax)

ax.set\_title(f'{col} Distribution')

ax.set\_xlabel(col)

ax.set\_ylabel('Density')

ax.axvline(ci[0], color='r', linestyle='--', label='90% Confidence Interval')

ax.axvline(ci[1], color='r', linestyle='--')

ax.legend()

plt.show()

**5- A new drilling rig took 24 days to complete a well. Was this a problematic well?**

def is\_problematic(df, column, value):

mu, std = norm.fit(df[column])

# Calculate the 90% confidence interval

ci = norm.interval(0.9, loc=mu, scale=std)

# Check if the value is within the confidence interval

if ci[0] <= value <= ci[1]:

print(f'The value {value} is within the 90% confidence interval [{ci[0]:.2f}, {ci[1]:.2f}]')

print("No")

problematic=False

else:

print(f'The value {value} is not within the 90% confidence interval [{ci[0]:.2f}, {ci[1]:.2f}]')

print("Yes")

problematic=True

return problematic

is\_problematic(df,'Days to Total Depth (d)’,24)

**6- A new drilling rig took 26 days to complete a well. Was this a problematic well?**

is\_problematic(df,'Days to Total Depth (d)',26)

**7- Load data for a new drilling rig**

column = 'Days to Total Depth (d)'

df\_new\_rig = pd.read\_csv('data/operations/days\_to\_total\_depth\_new\_rig.csv')

df\_new\_rig.head()

sns.histplot(df\_new\_rig[column], kde=True, bins=30)

# Calculate the 90% confidence interval for each KPI

rig\_mu, rig\_std = norm.fit(df\_new\_rig['Days to Total Depth (d)'])

rig\_ci = norm.interval(0.9, loc=rig\_mu, scale=rig\_std)

plt.title(f'{column} Distribution\nMean: {rig\_mu:.2f}, Standard Deviation: {rig\_std:.2f}')

# Plot the confidence interval for each KPI

plt.axvline(rig\_ci[0], color='r', linestyle='--', label='90% Confidence Interval')

plt.axvline(rig\_ci[1], color='r', linestyle='--')

plt.axvline(rig\_mu, color='g', linestyle='-', label='Mean')

plt.legend()

sns.histplot(df[column], kde=True, bins=30)

# Calculate the 90% confidence interval for each KPI

field\_mu, field\_std = norm.fit(df['Days to Total Depth (d)'])

field\_ci = norm.interval(0.9, loc=field\_mu, scale=field\_std)

plt.title(f'{column} Distribution\nMean: {field\_mu:.2f}, Standard Deviation: {field\_std:.2f}')

# Plot the confidence interval for each KPI

plt.axvline(field\_ci[0], color='r', linestyle='--', label='90% Confidence Interval')

plt.axvline(field\_ci[1], color='r', linestyle='--')

plt.axvline(field\_mu, color='g', linestyle='-', label='Mean')

plt.legend()

**8- Run statistical hypothesis testing to determine if the average of the new rig is not lower than the general average.**

from scipy.stats import norm

rig\_n = len(df\_new\_rig)

field\_n = len(df)

# Calculate the test statistic

z = (rig\_mu - field\_mu) / ((rig\_std\*\*2 / rig\_n) + (field\_std\*\*2 / field\_n))\*\*(1/2)

# Calculate the p-value

p\_value = norm.sf(z)

# Set the significance level

alpha = 0.05

# Calculate the critical value

critical\_value = norm.ppf(1 - alpha)

# Compare the test statistic to the critical value

if z > critical\_value:

print("Reject the null hypothesis - rig\_mu is not smaller than field\_mu")

else:

print("Fail to reject the null hypothesis - rig\_mu can be smaller than field\_mu")

# Create a range of z-scores

z\_range = np.arange(-3, 3, 0.01)

# Create a plot of the standard normal distribution

fig, ax = plt.subplots()

ax.plot(z\_range, norm.pdf(z\_range), 'b-', lw=2, alpha=0.6) # label='Standard Normal Distribution')

# Shade the area to the right of the critical value or the regection zone

ax.fill\_between(z\_range, 0, norm.pdf(z\_range), where=z\_range >= critical\_value, color='red', alpha=0.5, label='Regection Zone')

# Add a vertical line for the test statistic

ax.axvline(x=z, color='black', linestyle='--', lw=2, label='z-score')

# Add a vertical line for the critical value

ax.axvline(x=critical\_value, color='green', linestyle='--', lw=2, label='Critical Value')

# Add labels and a legend

ax.set\_xlabel('z-score')

ax.set\_ylabel('Probability Density')

ax.set\_title('One-Tailed Two-Sample z-Test')

ax.legend(loc='best')

# Show the plot

plt.show()

**9- Load and analyze data for this rig: days\_to\_total\_depth\_new\_rig\_1.csv**

**10- Load and analyze data for this rig: days\_to\_total\_depth\_new\_rig\_2.csv**

**11- Load and analyze performance data for after introducing new training**

column = 'Slip to Slip (min)'

df\_training = pd.read\_csv('data/operations/after\_training.csv')

df\_training.head()

sns.histplot(df\_training[column], kde=True, bins=30)

# Calculate the 90% confidence interval for each KPI

training\_mu, training\_std = norm.fit(df\_training[column])

training\_ci = norm.interval(0.9, loc=training\_mu, scale=training\_std)

plt.title(f'{column} Distribution\nMean: {training\_mu:.2f}, Standard Deviation: {training\_std:.2f}')

# Plot the confidence interval for each KPI

plt.axvline(training\_ci[0], color='r', linestyle='--', label='90% Confidence Interval')

plt.axvline(training\_ci[1], color='r', linestyle='--')

plt.axvline(training\_mu, color='g', linestyle='-', label='Mean')

plt.legend()

training\_n = len(df\_after\_training)

#H0: Xt - X = 0

#Ha: Xt - X < 0

alpha = 0.05

df = n + m - 2

cv = t.ppf(alpha, df)

t\_stat = (training\_mu - field\_mu) / math.sqrt((field\_std\*\*2/field\_n) + (training\_std\*\*2/training\_n))

print(f"After Training: mu {training\_mu:.2f}, std {training\_std:.2f}")

print(f"Field data: mu {field\_mu:.2f}, std {field\_std:.2f}")

if t\_stat < cv:

print("Reject the null hypothesis. Training and field time are not the same")

else:

print("Fail to reject the null hypothesis.Training and field time can be the same.")

# Plot the distribution and the rejection zone

x = np.linspace(-5, 5, 1000)

y = t.pdf(x, df)

plt.plot(x, y, 'b-', linewidth=2)

plt.axvline(x=cv, color='r', linestyle='--')

plt.fill\_between(x[x<=cv], y[x<=cv], color='r', alpha=0.5)

plt.axvline(x=t\_stat, color='g', linestyle='--')

plt.title('t-Distribution with Degrees of Freedom = ' + str(df))

plt.xlabel('t-Value')

plt.ylabel('Probability Density')

plt.show()