In [40]:

```
import pandas as pd
import numpy as np

dataset = pd.read_csv("intellicathdefense.csv")
dataset.head
```

Out[40]:

```
<bound method NDFrame.head of       urine_output  urine_flow_rate  catheter_bag_volume  \
0        97.131536        98.872967             0.000000
1        99.346882       100.763500             0.160032
2       101.285898        96.603298             0.320064
3       118.937229       121.835150             0.480096
4        85.055654        77.605240             0.640128
...            ...              ...                  ...
4995    107.756547       104.728205           799.359872
4996    127.547332       126.109911           799.519904
4997    107.567079       112.405974           799.679936
4998    134.270595       137.389061           799.839968
4999     67.601604        67.679527           800.000000

      remaining_catheter_bag_volume  time
0                        800.000000   720
1                        799.839968   719
2                        799.679936   719
3                        799.519904   719
4                        799.359872   719
...                             ...   ...
4995                       0.640128     0
4996                       0.480096     0
4997                       0.320064     0
4998                       0.160032     0
4999                       0.000000     0

[5000 rows x 5 columns]>
```

In [42]:

```
X = dataset[[
    "urine_output",
    "urine_flow_rate",
    "catheter_bag_volume",
    "remaining_catheter_bag_volume"
]]
```

In [43]:

```
dataset.loc[dataset["catheter_bag_volume"] == 0, "time"] = 720
dataset.loc[dataset["catheter_bag_volume"] >= 800, "time"] = 0

y = dataset["time"]
```

In [44]:

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_poly)
```

In [45]:

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_decimal, test_size=0.2, random_state=42)

model_1 = Ridge(alpha=1.0)
model_1.fit(X_train, y_train)
```

Out[45]:

```
▾ Ridge  i ?

Ridge()
```

In [46]:

```
y_pred = model_1.predict(X_test)
```

In [52]:

```
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("R2 Score (Test Set):", r2)
```

```
Mean Squared Error (MSE): 2.4447859401808627e-05
Mean Absolute Error (MAE): 0.004188907855012891
R2 Score (Test Set): 0.9999980486347136
```

In [53]:

```
from joblib import dump

dump(model_1, 'model_1.joblib')
```

Out[53]:

```
['model_1.joblib']
```

In [54]:

```
from joblib import load

model_loaded = load('model_1.joblib')

prediction = model_loaded.predict(X_test)
```

In [55]:

```
import pickle

with open("model_1.pkl", "wb") as file:
    pickle.dump(model_1, file)

# Save the polynomial features and scaler for future use
with open("polynomial_features.pkl", "wb") as file:
    pickle.dump(poly, file)

with open("scaler.pkl", "wb") as file:
    pickle.dump(scaler, file)

model = pickle.load(open('model_1.pkl', 'rb'))
```

In [ ]:

In [ ]: