In [15]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import pickle
```

In [16]:
```
file_path = "final.csv"
df = pd.read_csv(file_path)
df.head
```

Out[16]:
```
<bound method NDFrame.head of       Unnamed: 0  urine_output  urine_flow_rate  catheter_bag_volume  \
0              0     97.131536        98.872967             0.000000
1              1     99.346882       100.763500             0.266756
2              2    101.285898        96.603298             0.533511
3              3    118.937229       121.835150             0.800267
4              4     85.055654        77.605240             1.067022
...          ...           ...              ...                  ...
2995        2995    119.699002       113.404649           798.932978
2996        2996     84.333825        86.112174           799.199733
2997        2997    115.705107       119.858386           799.466489
2998        2998     77.013908        76.538159           799.733244
2999        2999    120.628318       126.819734           800.000000

      remaining_catheter_bag_volume       time
0                        800.000000  720.00000
1                        799.839968  720.00000
2                        799.679936  720.00000
3                        799.519904  720.00000
4                        799.359872  719.03968
...                             ...        ...
2995                     320.704141    0.96032
2996                     320.544109    0.72024
2997                     320.384077    0.48016
2998                     320.224045    0.24008
2999                     320.064013    0.00000

[3000 rows x 6 columns]>
```

In [17]:
```
df = df.sample(frac=1, random_state=42).reset_index(drop=True)
```

In [18]:
```
df
```

Out[18]:

	Unnamed: 0	urine_output	urine_flow_rate	catheter_bag_volume	remaining_catheter_bag_volume	time
0	1801	100.288331	99.708338	480.426809	511.782356	287.615872
1	1190	109.741731	108.960005	317.439146	609.561912	434.304768
2	1817	130.575577	132.636680	484.694898	509.221844	283.774591
3	251	106.682525	112.519478	66.955652	759.831966	659.739913
4	2505	125.862961	132.296956	668.222741	399.119824	118.599533
...	...	...	...	...	...	...
2995	1638	82.352214	86.016423	436.945649	537.867574	326.748916
2996	1095	81.833175	84.793334	292.097366	624.764953	457.112371
2997	1130	102.872617	97.986272	301.433811	619.163833	448.709570
2998	1294	71.062341	72.176661	345.181727	592.918584	409.336446
2999	860	139.106524	138.966604	229.409803	662.372474	513.531177

3000 rows × 6 columns

In [19]:
```
df["bag_full_signal"] = (df["catheter_bag_volume"] >= 800).astype(int)

np.random.seed(42)
df["catheter_bag_volume"] += np.random.normal(loc=0, scale=5, size=len(df)) #noise
```

In [20]:
```
X = df[["catheter_bag_volume", "bag_full_signal"]]  # features
y = df["time"]
```

In [21]:
```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42, shuffle=True)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42, shuffle=True)
```

In [22]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train[["catheter_bag_volume"]])
X_val_scaled = scaler.transform(X_val[["catheter_bag_volume"]])
X_test_scaled = scaler.transform(X_test[["catheter_bag_volume"]])
```

In [23]:
```
X_train_scaled = np.column_stack((X_train_scaled, X_train["bag_full_signal"].values))
X_val_scaled = np.column_stack((X_val_scaled, X_val["bag_full_signal"].values))
X_test_scaled = np.column_stack((X_test_scaled, X_test["bag_full_signal"].values))
```

In [24]:
```
model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

Out[24]:

▾ LinearRegression

```
LinearRegression()
```

In [25]:
```
y_val_pred = model.predict(X_val_scaled)
val_mae = mean_absolute_error(y_val, y_val_pred)
val_mse = mean_squared_error(y_val, y_val_pred)
val_rmse = np.sqrt(val_mse)
val_r2 = r2_score(y_val, y_val_pred)

print("\nMODEL PERFORMANCE ON VALIDATION SET:")
print(f"Validation MAE: {val_mae:.4f}")
print(f"Validation MSE: {val_mse:.4f}")
print(f"Validation RMSE: {val_rmse:.4f}")
print(f"Validation R² Score: {val_r2:.4f}")
```

```
MODEL PERFORMANCE ON VALIDATION SET:
Validation MAE: 3.4140
Validation MSE: 18.4875
Validation RMSE: 4.2997
Validation R² Score: 0.9996
```

In [26]:
```
y_test_pred = model.predict(X_test_scaled)
test_mae = mean_absolute_error(y_test, y_test_pred)
test_mse = mean_squared_error(y_test, y_test_pred)
test_rmse = np.sqrt(test_mse)
test_r2 = r2_score(y_test, y_test_pred)

print("\nMODEL PERFORMANCE ON TEST SET:")
print(f"Test MAE: {test_mae:.4f}")
print(f"Test MSE: {test_mse:.4f}")
print(f"Test RMSE: {test_rmse:.4f}")
print(f"Test R² Score: {test_r2:.4f}")
```

```
MODEL PERFORMANCE ON TEST SET:
Test MAE: 3.5993
Test MSE: 20.6502
Test RMSE: 4.5442
Test R² Score: 0.9995
```

In [27]:
```
with open("linear_regression_model3.pkl", "wb") as file:
    pickle.dump(model, file)
with open("scaler_3.pkl", "wb") as file:
    pickle.dump(scaler, file)
```

In [ ]: