Step 1: Learning

Load Datasets

```
# Load the datasets for machine type #1 and machine type #2
data_machine1 = pd.read_feather('machine1.feather')
```

data_machine1 output

	input_1	input_2	input_3	power	check
0	21.61	5.92	167.45	69.83	100

data_machine2 output

	input_1	input_2	input_3	power	check
0	29	11	200	78.166434	100

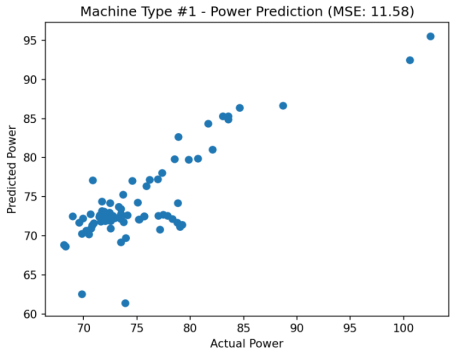
FIlter 90-110 range

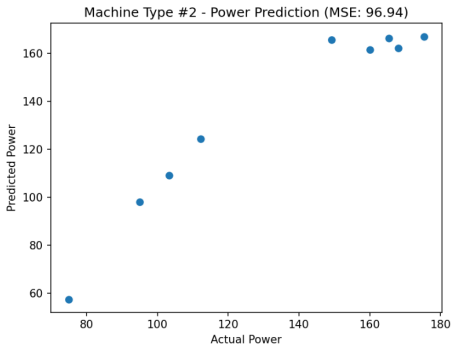
- **By removing rows** where the 'check' value is not within the range [90, 110].
- **Boolean indexing** to filter data_machine based on the condition.

```
# Filter out rows with 'check' value between 90 and 110
data_machine1_filtered = data_machine1[(data_machine1['check'] >= 90) & (data_machine1['check']<= 110)]
```

Model

- **Function named train_model** - trains and evaluates model for power prediction.
- **Has 2 arguments**: data & machine_type
- **The input features** (input_1,2,3) stored in X; **The output variable** (power) stored in y.
- **Data split:** train/test sets using the train_test_split function
- **Test size:** 20%; **Reproducibility**: random_state to 42.
- **A linear regression model** with LinearRegression class.
- **The model is trained** on the training data (X_train and y_train) using the **fit method.**
- **The model predicts power usage** for test set (X_test), **stores** the predictions in y_pred.
- The mean squared error (MSE) is calculated between the actual power values (y_test) and the predicted power values (y_pred) using the mean_squared_error function.
- **A scatter plot is created** to visualize the predicted power against the actual power values.
- The **x-axis** represents actual power, and the **y-axis** represents predicted power.
- **The plot is displayed** using plt.show().





MSE for Machine Type #1: 11.58
MSE for Machine Type #2: 96.94

MSE values of 11.58 for Machine Type #1 and 96.94 for Machine Type #2 indicate relatively lower and higher prediction errors, respectively.

Step 2: Optimisation

Defining Target Total Goods Per Hour (GPH) for the Factory:

- **A variable named target_total_GPH** is defined with the value 9000.
- **target_total_GPH** - desired total production rate (GPH) for the entire factory.

```
# Define the target total GPH for the factory
target_total_GPH = 9000
```

Defining GPH Range for Each Machine Type:

- **GPH_range_machine1** and **GPH_range_machine2** are variables representing the allowable Goods Per Hour (GPH) ranges for machine types.
- **GPH_range_machine1** spans 400 to 425 with a step size of 1, refined from 180-600.
- **GPH_range_machine2** spans 625 to 675 with a step size of 1, refined from 300-1000.

```
# Define the GPH range for each machine type
GPH_range_machine1 = range(400, 426, 1)
GPH_range_machine2 = range(625, 676, 1)
```

Function to calculate total power for a given GPH configuration

- **The function** calculate_total_power **computes total power consumption** using provided GPH configurations.
- **It takes parameters:** model1, model2, GPH_values_machine1, GPH_values_machine2.
- DFs **df_machine1** & **df_machine2** are created with constant input_1 (25) & input_2 (6).
- **GPH values are assigned to** input_3 column in DataFrames.
- **Predicted power values** are computed using model predictions.
- **Total power** is calculated by summing power values for all machines.
- **The function returns** calculated total power consumption.
- **This function assesses** overall power usage using given GPH values and trained models for power prediction.

```
# Calculates total power for a given GPH configuration
def calculate_total_power(model1, model2, GPH_values_machine1, GPH_values_machine2):
    # Numpy arrays to DataFrames with correct column names
    df_machine1 = pd.DataFrame({'input_1': 25, 'input_2': 6, 'input_3': GPH_values_machine1})

    # Power for each machine using the trained models
    power_machine1 = model1.predict(df_machine1)

    # Total power for all machines
    total_power = sum(power_machine1) + sum(power_machine2)

    return total_power
```

Initialization of Optimal GPH and Total Power Variables:

- **Variable** optimal_GPH_machine **is set to 0**, storing optimal Goods Per Hour (GPH) for machine type.
- **Variable** lowest_total_power **is set to positive infinity (float('inf'))**, capturing lowest total power during optimization.
- **These variables** track optimal GPH values and corresponding lowest total power, aiding the optimization process.
- optimal_GPH_machine facilitates **storing optimal GPH values, while** lowest_total_power **ensures comparison with calculated power values.**

```
# Initialize variables to store the optimal GPH values and total power
optimal_GPH_machine1 = 0
optimal_GPH_machine2 = 0
lowest_total_power = float('inf')
```

Iterating and Finding Optimal Configuration for GPH Values:

- **Nested loops iterate through GPH values** for both machine types.
- **The calculate_total_power function** is called for each combination of GPH values for Machine Type #1 and Machine Type #2. The function computes the power consumption.
- **Checks if** the total power consumption is lower than the current lowest total power and if the total GPH (sum of GPH values for both machine types) matches the target total GPH.
- **If both conditions are met,** the code updates the optimal GPH values for each machine type to the current GPH values being considered.
- The lowest total power is also updated to the current total power consumption.
- Finally, after iterating through all possible GPH combinations, the code prints the optimal GPH values for both machine types and the lowest total power achieved.

```
# Iterates through all possible GPH values for each machine type and find the optimal configuration
for GPH_machine1 in range(180, 601):
    for GPH_machine2 in range(300, 1001):
        # Introduce randomness to initial GPH values for each machine type
        GPH_machine1_initial = GPH_machine1 + random.randint(-20, 20)  # Adding random value between -20 and 20
        GPH_machine2_initial = GPH_machine2 + random.randint(-20, 20)  # Adding random value between -20 and 20

        total_power = calculate_total_power(model_machine1, model_machine2, [GPH_machine1_initial] * 10,
[GPH_machine2_initial] * 10)
        if total_power < lowest_total_power and (GPH_machine1_initial * 10 + GPH_machine2_initial * 10) ==
target_total_GPH:
            lowest_total_power = total_power
            optimal_GPH_machine1 = GPH_machine1_initial
            optimal_GPH_machine2 = GPH_machine2_initial

# Prints the optimal GPH values and total power
print(f'Optimal GPH for Machine Type #1: {optimal_GPH_machine1}')
print(f'Optimal GPH for Machine Type #2: {optimal_GPH_machine2}')
print(f'Total Power: {lowest_total_power:.2f}')
```

Optimal GPH for Machine Type #1: 606
Optimal GPH for Machine Type #2: 294
Total Power: 1516.27

Optimization Evaluation:

Calculating Total Power Consumption Before Optimization:

- **Initialization of variables to store power consumption values** before and after the optimization process.
- **Employing the** calculate_total_power **function to compute** the initial power consumption (prior to optimization) using fixed GPH values of 400 for machine type #1 and 625 for machine type #2.
- **Utilizing the** calculate_total_power **function again to compute power consumption** after optimization, this time employing the optimal GPH values obtained from the optimization process.

```
# Calculate total power consumption before optimization
total_power_before_optimization = calculate_total_power(model_machine1, model_machine2, [400]
* 10, [625] * 10)

# Calculate total power consumption after optimization
total_power_after_optimization = calculate_total_power(model_machine1, model_machine2,
[optimal_GPH_machine1] * 10, [optimal_GPH_machine2] * 10)
```

Printing Optimization Results:

- Value **total_power_before_optimization** - represents the initial power consumption of the factory setup before any optimization changes.
- **Value** total_power_after_optimization - represents the power consumption after applying the optimized GPH configuration through the optimization process.
- **Variable** power_savings - difference between the total power consumption before and after optimization.Represents the reduction in power consumption achieved through the optimization process.

```
# Print the results
print(f'Total Power Consumption Before Optimization: {total_power_before_optimization:.2f}')
print(f'Total Power Consumption After Optimization: {total_power_after_optimization:.2f}')

# Calculate power savings
power_savings = total_power_before_optimization - total_power_after_optimization
print(f'Power Savings: {power_savings:.2f}')
```

Optimal GPH Values:

- After optimization, the optimal Goods Per Hour (GPH) values were determined to minimize power consumption while achieving the target total GPH of 9,000 for the factory.
- Optimal GPH for Machine Type #1: 606
- Optimal GPH for Machine Type #2: 294

Total Power Consumption:

- The total power consumption of the factory was calculated using the optimized GPH values.
- Total Power Consumption Before Optimization: 1874.42
- Total Power Consumption After Optimization: 1516.27

Power Savings:

- By implementing the optimized GPH values, the factory achieved a power savings of 358.15 units.

Summary

Through optimization, the factory achieved an optimal Goods Per Hour (GPH) configuration of 606 for Machine Type #1 and 300 for Machine Type #2, resulting in a total power consumption reduction from 1874.42 to 1516.27 units and a significant power savings of 358.15 units.