CISC3024 Machine Learning Final Project

- Title: Wound Detection
- Groupmates: Huang Yanzhen DC126732, Yang Zhihan DC127992

In [157…
```python
# Basics
import os
import random
import cv2
import copy
import time
from itertools import product
from typing import List, Callable, Any, Union

# Pre-processing
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.lines as mlines
from PIL import Image
import numpy as np
from sklearn.metrics import mean_squared_error
import pandas as pd

# Model Training
import pickle
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.model_selection import KFold
```

In [2]:
```python
model_path = "./models"
```

1. Dataset

1.1 Data Retrieval

In [3]:
```python
def get_labels(data_type: str) -> np.ndarray:
    """
    Get ground truth labels from .csv file.
    :param data_type: Type of data: Training or Testing.
    """
    df = pd.read_csv(f"./Wound/{data_type}/myData.csv", delimiter=";")
    return df.to_numpy()
```

In [46]:
```python
def get_images(data_type: str,
               image_names: np.ndarray,
               augmentation: Union[Callable[[np.ndarray, Any], np.ndarray], None] = None,
               flatten=True,
               resize=True,
               **kwargs) -> np.ndarray:
    """
    Get the images from directory.
    :param data_type: Type of data: Training or Testing.
    :param image_names: Names of images from ground truth.
    :param augmentation: Augmentation function.
    :param flatten: Whether to flatten the images.
    :param kwargs: Other arguments to pass to augmentation function.
    """
    images = []
    for i_name in image_names:
        img = Image.open(os.path.join(f"./Wound/{data_type}/", i_name))
        if resize:
            img = img.resize((32, 32), Image.BICUBIC)
        img = np.array(img)
        if augmentation:
            img = augmentation(img, **kwargs)
        images.append(img.flatten() if flatten else img)

    images = np.array(images)

    return images
```

1.2 Image Augmentation

In [5]:
```python
def add_black_edge(img: np.array, w: int = 4) -> np.array:
    """
    Image augmentation. Add an inner black edge to an image.
    :param img: Image to be processed.
    :param w: Width of the edge.
    """
    if w > min(img.shape[0:2]) // 2:
        raise ValueError("Width of the edge must be smaller than half of the shorter side of an image.")

    new_img = np.zeros_like(img)
    new_img[w:-w, w:-w, :] = img[w:-w, w:-w, :]
    return new_img
```

In [6]:
```python
def stretch(img: np.ndarray, f: List[float]) -> np.ndarray:
    """
    Image augmentation. Stretch an image on the width and height side.
    :param img: Image to be augmented.
    :param f: Factor tuple. Width and Height.
    """
    fw, fh = f
    if fw < 1 or fh < 1:
        raise ValueError("Width and height factors should be greater than or equal to 1.")

    # New widths
    new_width = int(img.shape[1] * fw)
    new_height = int(img.shape[0] * fh)

    # Adjust image
    img_pil = Image.fromarray(img)
    img_resized = img_pil.resize((new_width, new_height), Image.BICUBIC)

    # Crop regions
    # Keep 32x32 size
    left = (new_width - 32) // 2
    top = (new_height - 32) // 2
    right = left + 32
    bottom = top + 32

    # Crop image
    img_cropped = img_resized.crop((left, top, right, bottom))

    # Convert to numpy array
    img_stretched = np.array(img_cropped)

    return img_stretched
```

2. Train Model

One trainig of model would result in the following structure:

```
{
    "x": {
        "Best MSE": smallest_mse,
        "Best Fold": best_fold_idx,
        "Avg MSE": avg_mse,
        "model": ModelInstance,
    },
    "y": {
        "Best MSE": smallest_mse,
        "Best Fold": best_fold_idx,
        "Avg MSE": avg_mse,
        "model": ModelInstance,
    },
    "w": {
        "Best MSE": smallest_mse,
        "Best Fold": best_fold_idx,
        "Avg MSE": avg_mse,
        "model": ModelInstance,
    },
    "h": {
        "Best MSE": smallest_mse,
        "Best Fold": best_fold_idx,
        "Avg MSE": avg_mse,
        "model": ModelInstance,
    },
}
```

This is named as an "experiment object".

In [195…
```python
def train(ModelInstance, X, Y, desc: str = "DESC", n_fold: int = 3, save: bool = False):
    """
    Train the model. Output would be of shape:
    :param ModelInstance: Instance of a model class.
    :param X: Image data.
    :param Y: Image file name, x, y, w, h.
    :param desc: Description of the saved file.
    :param n_fold: Number of folds for cross-validation.
    :param save: Whether to save the experiment object.
    """
    model_name = ModelInstance.__class__.__name__
    semantic_y = ["File Name", "x", "y", "w", "h"]

    # Print Model configurations
    print(f"Training model {model_name}. Description: {desc}\nStarted at: {time.time()}")
    # Predict all for x, y, w, h
    exp = {}
    for i in range(1, Y.shape[1]):
        # Totally 4 labels to predict.
        # Select one of them.
        y = Y[:, i]

        # Split original data into 3 parts
        # to perform cross-validation
        kf = KFold(n_splits=n_fold, shuffle=True, random_state=1919810)
        splits = kf.split(X)

        # Record MSE of each fold.
        # Keep the model with the smallest MSE
        mse_scores = []
        cur_best_model = None
        cur_smallest_MSE = np.inf
        for train_index, val_index in splits:
            X_train, X_val = X[train_index], X[val_index]
            y_train, y_val = y[train_index], y[val_index]

            model = copy.deepcopy(ModelInstance)
            model.fit(X_train, y_train)

            y_pred = model.predict(X_val)
            mse = mean_squared_error(y_val, y_pred)
            mse_scores.append(mse)

            if cur_smallest_MSE > mse_scores[-1]:
                cur_best_model = copy.deepcopy(model)

        exp[semantic_y[i]] = {
            "Best MSE": cur_smallest_MSE,
            "Best Fold": np.argmin(mse_scores),
            "Avg MSE": np.mean(mse_scores),
            "model": copy.deepcopy(cur_best_model)
        }
        del cur_best_model

        print(f"{semantic_y[i]} - Avg MSE={np.mean(mse_scores):.4f}, "
              f"Best MSE={np.min(mse_scores):.4f} at index {np.argmin(mse_scores)}")

    # Save models
    print(f"Ended at {time.time()}\n\n")
    if save:
        time_str = str(time.time()).replace(".", "")
        pickle.dump(exp, open(f"./save_models/{model_name}_{desc}_{time_str}.sav", "wb"))
    return exp
```

In [22]:
```python
def grid_search(ModelClass, hyper_params, hyper_param_names, kwarg_names, **kwargs):
    """
    Perform grid search for the given model class and hyperparameters.
    :param ModelClass: The class to instantiate the model.
    :param hyper_params: The product of two lists of candidate hyperparameters.
    :hyper_param_names: The display name of two hyperparameters.
    :kwarg_names: Names of keyword arguments to be passed into the model.
    :returns: A list of dictionaries containing hyperparameters and experiment objects.
    """
    param_exps = []

    n1, n2 = hyper_param_names
    kw1, kw2 = kwarg_names

    model_name = ModelClass.__name__

    # Data Augmentation
    # Change some useless information
    Y_ori = get_labels(data_type="Training")
```

```python
    X_ori = get_images(data_type="Training", image_names=Y_ori[:, 0])

    # Add black edge
    Y_be = get_labels(data_type="Training")
    X_be = get_images(data_type="Training", image_names=Y_be[:, 0], augmentation=add_black_edge, w=4)

    # Stretch height
    Y_sh = get_labels(data_type="Training")
    X_sh = get_images(data_type="Training", image_names=Y_sh[:, 0], augmentation=stretch, f=[1.0, 1.05])
    Y_sh[:, 4] *= 1.05

    # Stretch Width
    Y_sw = get_labels(data_type="Training")
    X_sw = get_images(data_type="Training", image_names=Y_sw[:, 0], augmentation=stretch, f=[1.05, 1.0])
    Y_sw[:, 3] *= 1.05

    X = np.concatenate((X_ori, X_be, X_sh, X_sw))
    Y = np.concatenate((Y_ori, Y_be, Y_sh, Y_sw))

    for param1, param2 in hyper_params:
        # Dynamically add the hyperparameters to kwargs
        model_kwargs = {
            kw1: param1,
            kw2: param2
        }

        model_kwargs.update(kwargs)

        model_instance = ModelClass(**model_kwargs)
        exp = train(model_instance, X, Y, desc=f"{n1}-{param1}--{n2}-{param2}", n_fold=3, save=False)
        param_exps.append({
            n1: param1,
            n2: param2,
            "exp": exp
        })

    time_str = str(time.time()).replace(".", "")
    pickle.dump(param_exps, open(os.path.join(model_path, f"{model_name}_{n1}-{n2}_{time_str}.sav"), "wb"))
    return param_exps
```

In [189…
```python
def test(exp_list, Y_test, X_test):
    """
    Test the model's performance with test dataset.
    :param exp_list: List of experiment objects loaded from local files.
    :param Y_test: Image file name, x, y, w, h.
    :param X_test: Image data.
    """
    # Y_test = get_labels(data_type="Test")
    # X_test = get_images(data_type="Test", image_names=Y_test[:,0])
    results = []
    for i, exp in enumerate(exp_list):
        param_name1, param_name2, _ = exp.keys()
        param1, param2, models = list(exp.values())

        model_x, model_y, model_w, model_h = (models["x"]["model"],
                                              models["y"]["model"],
                                              models["w"]["model"],
                                              models["h"]["model"])

        y_x, y_y, y_w, y_h = Y_test[:, 1], Y_test[:, 2], Y_test[:, 3], Y_test[:, 4]

        y_pred_x, y_pred_y, y_pred_w, y_pred_h = (model_x.predict(X_test),
                                                  model_y.predict(X_test),
                                                  model_w.predict(X_test),
                                                  model_h.predict(X_test))

        mse_x, mse_y, mse_w, mse_h = (mean_squared_error(y_x, y_pred_x),
                                      mean_squared_error(y_y, y_pred_y),
                                      mean_squared_error(y_w, y_pred_w),
                                      mean_squared_error(y_h, y_pred_h))

        weighted_avg_mse = (mse_x + mse_y) * 0.3 + (mse_w + mse_h) * 0.2

        results.append({
            param_name1: param1,
            param_name2: param2,
            "weighted_avg_mse": weighted_avg_mse
        })
    return results
```

In [190…
```python
def inference(model_dict, X):
    """
    Model inference on a single data.
```

```python
    :param model_dict: Model dictionary of the four models for x, y, w, h.
    :param X: The single image data.
    """
    model_x, model_y, model_w, model_h = (model_dict["x"],
                                          model_dict["y"],
                                          model_dict["w"],
                                          model_dict["h"])
    y_pred_x, y_pred_y, y_pred_w, y_pred_h = (model_x.predict(X),
                                              model_y.predict(X),
                                              model_w.predict(X),
                                              model_h.predict(X))
    return {
        "x": float(y_pred_x[0]),
        "y": float(y_pred_y[0]),
        "w": float(y_pred_w[0]),
        "h": float(y_pred_h[0])
    }
```

3. Grid Search

3.1 Grid Search: RandomForestRegressor

Hyper parameters for Random Forest Regressor:

- `nest` : `n_estimators` , Number of estimators.
- `maxd` : `max_depth` , Maximum Depth of the tree.
- `mins` : `min_samples_split` , Minimum sample number that allows a leaf to be split again.
- `minl` : `min_samples_leaf` , Minimum sample number a leaf requires.

In [175…

```python
rfr_nest = [10, 20, 30, 40, 50]
rfr_maxd = [11, 13, 15, 17, 19]
rfr_mins = [4, 6, 8, 10, 12]
rfr_minl = [6, 8, 10, 12, 14]
rfr_grid0 = product(rfr_nest, rfr_maxd)
rfr_grid1 = product(rfr_mins, rfr_minl)
```

3.1.1 Grid Search 0: Num Estimators + Max Deapth

In [9]:

```python
rfr_grid0_exp = grid_search(ModelClass=RandomForestRegressor,
                            hyper_params=rfr_grid0,
                            hyper_param_names=["nest", "maxd"],
                            kwarg_names=["n_estimators", "max_depth"])
```

```
Training model RandomForestRegressor. Description: nest-10--maxd-11
Started at: 1732799242.455379
x - Avg MSE=265.0281, Best MSE=226.6704 at index 1
y - Avg MSE=319.6748, Best MSE=206.1377 at index 1
w - Avg MSE=981.5292, Best MSE=914.7635 at index 0
h - Avg MSE=1024.7640, Best MSE=774.9044 at index 0
Ended at 1732799277.573622

Training model RandomForestRegressor. Description: nest-10--maxd-13
Started at: 1732799303.3531008
x - Avg MSE=303.6308, Best MSE=264.3288 at index 2
y - Avg MSE=330.7782, Best MSE=254.7151 at index 1
w - Avg MSE=1046.2773, Best MSE=935.1649 at index 0
h - Avg MSE=1015.2763, Best MSE=801.2331 at index 1
Ended at 1732799339.3567605

Training model RandomForestRegressor. Description: nest-10--maxd-15
Started at: 1732799364.6094503
x - Avg MSE=300.3605, Best MSE=246.8027 at index 2
y - Avg MSE=316.6453, Best MSE=246.9491 at index 1
w - Avg MSE=1039.2064, Best MSE=971.3801 at index 0
h - Avg MSE=1130.1478, Best MSE=980.0463 at index 1
Ended at 1732799401.170166

Training model RandomForestRegressor. Description: nest-10--maxd-17
Started at: 1732799426.1358972
x - Avg MSE=300.4805, Best MSE=221.9252 at index 2
y - Avg MSE=338.9396, Best MSE=276.1063 at index 1
w - Avg MSE=1021.4995, Best MSE=950.0199 at index 0
h - Avg MSE=1006.8791, Best MSE=919.2143 at index 2
Ended at 1732799462.3547573

Training model RandomForestRegressor. Description: nest-10--maxd-19
Started at: 1732799486.7901278
x - Avg MSE=302.8487, Best MSE=288.1919 at index 1
```

```
y - Avg MSE=328.5022, Best MSE=271.5901 at index 1
w - Avg MSE=1002.2346, Best MSE=857.0157 at index 0
h - Avg MSE=958.0220, Best MSE=848.5257 at index 1
Ended at 1732799523.2739859

Training model RandomForestRegressor. Description: nest-20--maxd-11
Started at: 1732799548.2014005
x - Avg MSE=285.0670, Best MSE=266.7257 at index 1
y - Avg MSE=309.2617, Best MSE=230.4450 at index 1
w - Avg MSE=812.8412, Best MSE=782.5403 at index 0
h - Avg MSE=876.8053, Best MSE=744.5018 at index 0
Ended at 1732799618.3518734

Training model RandomForestRegressor. Description: nest-20--maxd-13
Started at: 1732799642.5200229
x - Avg MSE=252.7599, Best MSE=221.1697 at index 2
y - Avg MSE=298.5948, Best MSE=219.2555 at index 1
w - Avg MSE=881.0838, Best MSE=758.1520 at index 2
h - Avg MSE=787.2464, Best MSE=742.0857 at index 0
Ended at 1732799715.4098089

Training model RandomForestRegressor. Description: nest-20--maxd-15
Started at: 1732799739.9309459
x - Avg MSE=253.8009, Best MSE=203.6632 at index 2
y - Avg MSE=289.7559, Best MSE=216.9240 at index 1
w - Avg MSE=936.1498, Best MSE=894.1271 at index 0
h - Avg MSE=913.3963, Best MSE=819.3239 at index 1
Ended at 1732799813.0232718

Training model RandomForestRegressor. Description: nest-20--maxd-17
Started at: 1732799838.0781047
x - Avg MSE=253.8418, Best MSE=222.3508 at index 2
y - Avg MSE=329.0912, Best MSE=246.1059 at index 1
w - Avg MSE=884.3192, Best MSE=718.6947 at index 0
h - Avg MSE=902.8129, Best MSE=831.9966 at index 2
Ended at 1732799910.988647

Training model RandomForestRegressor. Description: nest-20--maxd-19
Started at: 1732799935.958038
x - Avg MSE=268.1530, Best MSE=228.2566 at index 2
y - Avg MSE=329.3778, Best MSE=236.9652 at index 1
w - Avg MSE=849.3705, Best MSE=713.9744 at index 0
h - Avg MSE=901.6856, Best MSE=797.9166 at index 0
Ended at 1732800008.8168418

Training model RandomForestRegressor. Description: nest-30--maxd-11
Started at: 1732800033.754099
x - Avg MSE=272.9434, Best MSE=214.3267 at index 2
y - Avg MSE=309.8040, Best MSE=244.9138 at index 1
w - Avg MSE=874.0065, Best MSE=786.2825 at index 0
h - Avg MSE=892.1854, Best MSE=776.4926 at index 0
Ended at 1732800138.805168

Training model RandomForestRegressor. Description: nest-30--maxd-13
Started at: 1732800162.8637316
x - Avg MSE=274.6289, Best MSE=240.6389 at index 2
y - Avg MSE=293.3156, Best MSE=201.7539 at index 1
w - Avg MSE=831.7885, Best MSE=747.1287 at index 0
h - Avg MSE=793.4268, Best MSE=752.4546 at index 0
Ended at 1732800271.1492708

Training model RandomForestRegressor. Description: nest-30--maxd-15
Started at: 1732800295.3863888
x - Avg MSE=253.0778, Best MSE=208.0780 at index 1
y - Avg MSE=301.5480, Best MSE=218.6238 at index 1
w - Avg MSE=940.0108, Best MSE=835.9889 at index 0
h - Avg MSE=874.5578, Best MSE=814.1522 at index 1
Ended at 1732800404.3143518

Training model RandomForestRegressor. Description: nest-30--maxd-17
Started at: 1732800428.4378633
x - Avg MSE=278.1840, Best MSE=251.4836 at index 2
y - Avg MSE=300.7216, Best MSE=214.8380 at index 1
w - Avg MSE=855.0443, Best MSE=826.7070 at index 0
h - Avg MSE=792.9134, Best MSE=750.9542 at index 0
Ended at 1732800538.099123

Training model RandomForestRegressor. Description: nest-30--maxd-19
Started at: 1732800562.0520394
x - Avg MSE=220.2197, Best MSE=207.4006 at index 2
y - Avg MSE=284.1861, Best MSE=210.9069 at index 1
w - Avg MSE=854.7060, Best MSE=785.5057 at index 0
h - Avg MSE=814.5918, Best MSE=776.2419 at index 2
```

```
Ended at 1732800671.8010006

Training model RandomForestRegressor. Description: nest-40--maxd-11
Started at: 1732800695.766269
x - Avg MSE=251.8448, Best MSE=215.4979 at index 1
y - Avg MSE=289.9733, Best MSE=226.2858 at index 1
w - Avg MSE=896.8800, Best MSE=767.0302 at index 0
h - Avg MSE=869.4700, Best MSE=844.8714 at index 0
Ended at 1732800835.5782795

Training model RandomForestRegressor. Description: nest-40--maxd-13
Started at: 1732800860.1936045
x - Avg MSE=259.2456, Best MSE=233.5999 at index 2
y - Avg MSE=292.2659, Best MSE=208.6475 at index 1
w - Avg MSE=824.1160, Best MSE=760.9130 at index 0
h - Avg MSE=806.5849, Best MSE=778.1615 at index 0
Ended at 1732801004.565523

Training model RandomForestRegressor. Description: nest-40--maxd-15
Started at: 1732801029.0618072
x - Avg MSE=244.2120, Best MSE=226.4907 at index 1
y - Avg MSE=291.8532, Best MSE=225.0804 at index 1
w - Avg MSE=868.4658, Best MSE=724.4030 at index 0
h - Avg MSE=781.5694, Best MSE=727.7959 at index 1
Ended at 1732801210.976922

Training model RandomForestRegressor. Description: nest-40--maxd-17
Started at: 1732801254.3320477
x - Avg MSE=234.3425, Best MSE=200.5500 at index 2
y - Avg MSE=286.3102, Best MSE=221.9404 at index 1
w - Avg MSE=862.7032, Best MSE=726.3788 at index 0
h - Avg MSE=865.5574, Best MSE=813.9664 at index 0
Ended at 1732801497.655775

Training model RandomForestRegressor. Description: nest-40--maxd-19
Started at: 1732801539.3561637
x - Avg MSE=242.0519, Best MSE=206.8850 at index 2
y - Avg MSE=293.3265, Best MSE=227.6444 at index 1
w - Avg MSE=868.8398, Best MSE=772.2479 at index 0
h - Avg MSE=881.9131, Best MSE=807.3104 at index 0
Ended at 1732801778.9195538

Training model RandomForestRegressor. Description: nest-50--maxd-11
Started at: 1732801819.7143903
x - Avg MSE=270.1062, Best MSE=244.6453 at index 1
y - Avg MSE=284.4660, Best MSE=222.7727 at index 1
w - Avg MSE=889.8490, Best MSE=768.3454 at index 0
h - Avg MSE=836.6008, Best MSE=804.7312 at index 0
Ended at 1732802107.0316658

Training model RandomForestRegressor. Description: nest-50--maxd-13
Started at: 1732802147.970842
x - Avg MSE=241.5700, Best MSE=213.1301 at index 2
y - Avg MSE=270.0202, Best MSE=190.1522 at index 1
w - Avg MSE=847.1258, Best MSE=703.7665 at index 0
h - Avg MSE=842.7185, Best MSE=816.7929 at index 0
Ended at 1732802444.4113607

Training model RandomForestRegressor. Description: nest-50--maxd-15
Started at: 1732802485.3644946
x - Avg MSE=253.3668, Best MSE=212.1847 at index 2
y - Avg MSE=285.6002, Best MSE=196.1240 at index 1
w - Avg MSE=826.6384, Best MSE=753.8100 at index 2
h - Avg MSE=823.7715, Best MSE=784.7041 at index 1
Ended at 1732802781.4783213

Training model RandomForestRegressor. Description: nest-50--maxd-17
Started at: 1732802823.8504105
x - Avg MSE=255.9907, Best MSE=225.8875 at index 2
y - Avg MSE=285.5330, Best MSE=213.0238 at index 1
w - Avg MSE=865.0635, Best MSE=841.2231 at index 0
h - Avg MSE=821.2326, Best MSE=761.4187 at index 2
Ended at 1732803122.3551664

Training model RandomForestRegressor. Description: nest-50--maxd-19
Started at: 1732803163.495206
x - Avg MSE=253.7331, Best MSE=223.8237 at index 1
y - Avg MSE=275.0223, Best MSE=196.7816 at index 1
w - Avg MSE=862.6752, Best MSE=767.1612 at index 0
h - Avg MSE=864.4260, Best MSE=791.0012 at index 0
Ended at 1732803463.0916922
```

In [102…

```
with open(os.path.join(model_path, "RandomForestRegressor_nest-maxd_17328034630916922.sav" ), "rb") as rfr_grid(
```

```
    rfr_grid0_exp_loaded = pickle.load(rfr_grid0_exp_f)
```

In [103…
```
Y_test = get_labels(data_type="Test")
X_test = get_images(data_type="Test", image_names=Y_test[:,0])
rfr_grid0_results = test(exp_list=rfr_grid0_exp_loaded, Y_test=Y_test, X_test=X_test)
```

In [104…
```
rfr_grid0_results
```

Out[104…
```
[{'nest': 10, 'maxd': 11, 'weighted_avg_mse': np.float64(1037.866711863164)},
 {'nest': 10, 'maxd': 13, 'weighted_avg_mse': np.float64(1117.2152023979122)},
 {'nest': 10, 'maxd': 15, 'weighted_avg_mse': np.float64(1277.2782813552622)},
 {'nest': 10, 'maxd': 17, 'weighted_avg_mse': np.float64(1105.740378375)},
 {'nest': 10, 'maxd': 19, 'weighted_avg_mse': np.float64(1210.9086421250008)},
 {'nest': 20, 'maxd': 11, 'weighted_avg_mse': np.float64(1041.9624518709904)},
 {'nest': 20, 'maxd': 13, 'weighted_avg_mse': np.float64(1131.6653485552738)},
 {'nest': 20, 'maxd': 15, 'weighted_avg_mse': np.float64(1081.842526424827)},
 {'nest': 20, 'maxd': 17, 'weighted_avg_mse': np.float64(1022.9588525584991)},
 {'nest': 20, 'maxd': 19, 'weighted_avg_mse': np.float64(1022.3114634687497)},
 {'nest': 30, 'maxd': 11, 'weighted_avg_mse': np.float64(1002.6387328184359)},
 {'nest': 30, 'maxd': 13, 'weighted_avg_mse': np.float64(978.7584204580128)},
 {'nest': 30, 'maxd': 15, 'weighted_avg_mse': np.float64(1027.4104595490564)},
 {'nest': 30, 'maxd': 17, 'weighted_avg_mse': np.float64(997.9180385277775)},
 {'nest': 30, 'maxd': 19, 'weighted_avg_mse': np.float64(977.6827556527779)},
 {'nest': 40, 'maxd': 11, 'weighted_avg_mse': np.float64(979.8591202866849)},
 {'nest': 40, 'maxd': 13, 'weighted_avg_mse': np.float64(1008.4079476063424)},
 {'nest': 40, 'maxd': 15, 'weighted_avg_mse': np.float64(995.2617179147078)},
 {'nest': 40, 'maxd': 17, 'weighted_avg_mse': np.float64(1008.430395293182)},
 {'nest': 40, 'maxd': 19, 'weighted_avg_mse': np.float64(1009.1554874687499)},
 {'nest': 50, 'maxd': 11, 'weighted_avg_mse': np.float64(1003.441016700693)},
 {'nest': 50, 'maxd': 13, 'weighted_avg_mse': np.float64(1026.1017976872838)},
 {'nest': 50, 'maxd': 15, 'weighted_avg_mse': np.float64(990.8895085709411)},
 {'nest': 50, 'maxd': 17, 'weighted_avg_mse': np.float64(979.7050705067662)},
 {'nest': 50, 'maxd': 19, 'weighted_avg_mse': np.float64(1065.0097557450013)}]
```

In [105…
```
rfr_grid0_mean = np.mean([res["weighted_avg_mse"] for res in rfr_grid0_results])
rfr_grid0_best = min(rfr_grid0_results, key=lambda x: x["weighted_avg_mse"])
print(f"Best:{rfr_grid0_best}\nDiff:{rfr_grid0_best["weighted_avg_mse"]-rfr_grid0_mean}")
```

```
Best:{'nest': 30, 'maxd': 19, 'weighted_avg_mse': np.float64(977.6827556527779)}
Diff:-66.33404769846572
```

3.1.2 Grid Search 1: Min Samples Split, Min Samples Leaf

In [63]:
```
rfr_grid1_exp = grid_search(ModelClass=RandomForestRegressor,
                            hyper_params=rfr_grid1,
                            hyper_param_names=["mins", "minl"],
                            kwarg_names=["min_samples_split", "min_samples_leaf"],
                            # Settled Parameters
                            n_estimators=30,
                            max_depth=19)
```

```
Training model RandomForestRegressor. Description: mins-4--minl-6
Started at: 1732856800.760463
x - Avg MSE=326.0675, Best MSE=280.6331 at index 1
y - Avg MSE=327.7088, Best MSE=258.5277 at index 1
w - Avg MSE=1074.9534, Best MSE=991.6896 at index 0
h - Avg MSE=1065.0379, Best MSE=880.3963 at index 0
Ended at 1732856888.760798


Training model RandomForestRegressor. Description: mins-4--minl-8
Started at: 1732856888.760947
x - Avg MSE=338.8816, Best MSE=293.7372 at index 2
y - Avg MSE=363.2080, Best MSE=275.7450 at index 1
w - Avg MSE=1130.0578, Best MSE=1012.6302 at index 0
h - Avg MSE=1093.0109, Best MSE=1006.7138 at index 2
Ended at 1732856968.4516952


Training model RandomForestRegressor. Description: mins-4--minl-10
Started at: 1732856968.451767
x - Avg MSE=362.3157, Best MSE=326.9200 at index 1
y - Avg MSE=390.4445, Best MSE=299.2935 at index 1
w - Avg MSE=1202.0076, Best MSE=1027.7504 at index 0
h - Avg MSE=1182.0632, Best MSE=1125.1176 at index 0
Ended at 1732857042.551244


Training model RandomForestRegressor. Description: mins-4--minl-12
Started at: 1732857042.551657
x - Avg MSE=387.2047, Best MSE=347.7490 at index 2
y - Avg MSE=457.0623, Best MSE=380.3171 at index 1
w - Avg MSE=1349.5902, Best MSE=1210.3508 at index 0
```

```
h - Avg MSE=1321.2614, Best MSE=1190.2178 at index 0
Ended at 1732857110.7122219


Training model RandomForestRegressor. Description: mins-4--minl-14
Started at: 1732857110.712325
x - Avg MSE=414.6892, Best MSE=378.5784 at index 1
y - Avg MSE=448.2827, Best MSE=374.3078 at index 1
w - Avg MSE=1494.3689, Best MSE=1260.2119 at index 0
h - Avg MSE=1376.1246, Best MSE=1088.9423 at index 0
Ended at 1732857174.1005042


Training model RandomForestRegressor. Description: mins-6--minl-6
Started at: 1732857174.100596
x - Avg MSE=308.1539, Best MSE=277.8514 at index 1
y - Avg MSE=333.9432, Best MSE=255.1904 at index 1
w - Avg MSE=999.9675, Best MSE=882.3105 at index 0
h - Avg MSE=1008.4652, Best MSE=964.5096 at index 2
Ended at 1732857262.3876858


Training model RandomForestRegressor. Description: mins-6--minl-8
Started at: 1732857262.388395
x - Avg MSE=335.1505, Best MSE=309.3420 at index 2
y - Avg MSE=357.5733, Best MSE=282.5742 at index 1
w - Avg MSE=1131.7024, Best MSE=1015.0260 at index 0
h - Avg MSE=1110.7058, Best MSE=1039.7166 at index 0
Ended at 1732857343.509079


Training model RandomForestRegressor. Description: mins-6--minl-10
Started at: 1732857343.50933
x - Avg MSE=384.0030, Best MSE=352.8803 at index 2
y - Avg MSE=399.1749, Best MSE=328.0118 at index 1
w - Avg MSE=1229.0135, Best MSE=1135.7809 at index 0
h - Avg MSE=1225.1177, Best MSE=1127.3294 at index 0
Ended at 1732857417.773901


Training model RandomForestRegressor. Description: mins-6--minl-12
Started at: 1732857417.774088
x - Avg MSE=382.8800, Best MSE=363.1657 at index 1
y - Avg MSE=432.5145, Best MSE=354.1876 at index 1
w - Avg MSE=1313.8871, Best MSE=1160.9318 at index 0
h - Avg MSE=1277.1958, Best MSE=1226.2863 at index 2
Ended at 1732857485.851247


Training model RandomForestRegressor. Description: mins-6--minl-14
Started at: 1732857485.851352
x - Avg MSE=428.1227, Best MSE=385.7580 at index 1
y - Avg MSE=452.1929, Best MSE=377.2100 at index 1
w - Avg MSE=1428.0822, Best MSE=1288.4722 at index 0
h - Avg MSE=1358.7429, Best MSE=1216.5172 at index 0
Ended at 1732857549.002858


Training model RandomForestRegressor. Description: mins-8--minl-6
Started at: 1732857549.002997
x - Avg MSE=306.8516, Best MSE=292.9354 at index 2
y - Avg MSE=327.1679, Best MSE=252.7554 at index 1
w - Avg MSE=1046.2740, Best MSE=917.5920 at index 0
h - Avg MSE=960.6163, Best MSE=889.5745 at index 0
Ended at 1732857637.7303722


Training model RandomForestRegressor. Description: mins-8--minl-8
Started at: 1732857637.730516
x - Avg MSE=360.6055, Best MSE=298.0623 at index 1
y - Avg MSE=366.3725, Best MSE=271.5181 at index 1
w - Avg MSE=1169.0414, Best MSE=1074.6156 at index 2
h - Avg MSE=1104.9641, Best MSE=1032.8115 at index 2
Ended at 1732857719.070292


Training model RandomForestRegressor. Description: mins-8--minl-10
Started at: 1732857719.070383
x - Avg MSE=355.6136, Best MSE=318.7152 at index 2
y - Avg MSE=394.2206, Best MSE=300.6892 at index 1
w - Avg MSE=1264.0395, Best MSE=1032.7116 at index 0
h - Avg MSE=1225.3672, Best MSE=1126.8638 at index 0
Ended at 1732857793.819081
```

```
Training model RandomForestRegressor. Description: mins-8--minl-12
Started at: 1732857793.819297
x - Avg MSE=389.9695, Best MSE=354.9138 at index 1
y - Avg MSE=430.5395, Best MSE=364.4108 at index 1
w - Avg MSE=1366.1019, Best MSE=1225.0655 at index 0
h - Avg MSE=1260.9887, Best MSE=1031.0237 at index 0
Ended at 1732857862.781905


Training model RandomForestRegressor. Description: mins-8--minl-14
Started at: 1732857862.782188
x - Avg MSE=430.4666, Best MSE=392.7796 at index 1
y - Avg MSE=466.1909, Best MSE=358.7826 at index 1
w - Avg MSE=1477.8265, Best MSE=1271.8621 at index 0
h - Avg MSE=1375.9559, Best MSE=1204.4168 at index 0
Ended at 1732857926.73861


Training model RandomForestRegressor. Description: mins-10--minl-6
Started at: 1732857926.738683
x - Avg MSE=299.8077, Best MSE=277.9532 at index 2
y - Avg MSE=340.3186, Best MSE=256.9503 at index 1
w - Avg MSE=1105.3671, Best MSE=987.2962 at index 2
h - Avg MSE=1047.4830, Best MSE=938.0438 at index 0
Ended at 1732858015.5977721


Training model RandomForestRegressor. Description: mins-10--minl-8
Started at: 1732858015.597964
x - Avg MSE=331.7604, Best MSE=303.7267 at index 2
y - Avg MSE=368.8866, Best MSE=270.4076 at index 1
w - Avg MSE=1190.1597, Best MSE=1055.7868 at index 0
h - Avg MSE=1051.5489, Best MSE=1004.2388 at index 0
Ended at 1732858094.962867


Training model RandomForestRegressor. Description: mins-10--minl-10
Started at: 1732858094.963012
x - Avg MSE=355.7016, Best MSE=322.3980 at index 2
y - Avg MSE=375.9516, Best MSE=307.6887 at index 1
w - Avg MSE=1236.8031, Best MSE=1109.8976 at index 2
h - Avg MSE=1217.0728, Best MSE=1066.2648 at index 0
Ended at 1732858168.1504462


Training model RandomForestRegressor. Description: mins-10--minl-12
Started at: 1732858168.150672
x - Avg MSE=412.5593, Best MSE=359.6623 at index 2
y - Avg MSE=415.5667, Best MSE=349.6455 at index 1
w - Avg MSE=1354.4408, Best MSE=1166.7235 at index 0
h - Avg MSE=1227.3570, Best MSE=1103.0337 at index 0
Ended at 1732858235.984776


Training model RandomForestRegressor. Description: mins-10--minl-14
Started at: 1732858235.98503
x - Avg MSE=410.9545, Best MSE=387.5067 at index 1
y - Avg MSE=456.5057, Best MSE=362.5386 at index 1
w - Avg MSE=1452.3753, Best MSE=1248.6713 at index 0
h - Avg MSE=1384.1886, Best MSE=1149.0275 at index 0
Ended at 1732858299.3219578


Training model RandomForestRegressor. Description: mins-12--minl-6
Started at: 1732858299.322016
x - Avg MSE=326.4791, Best MSE=298.2585 at index 1
y - Avg MSE=322.2471, Best MSE=246.9821 at index 1
w - Avg MSE=1024.0750, Best MSE=875.6955 at index 0
h - Avg MSE=1012.1551, Best MSE=953.1265 at index 0
Ended at 1732858387.542382


Training model RandomForestRegressor. Description: mins-12--minl-8
Started at: 1732858387.542625
x - Avg MSE=348.1009, Best MSE=329.1898 at index 2
y - Avg MSE=372.8824, Best MSE=272.7803 at index 1
w - Avg MSE=1175.6590, Best MSE=985.4241 at index 0
h - Avg MSE=1065.9766, Best MSE=954.0569 at index 0
Ended at 1732858467.993131
```

```
Training model RandomForestRegressor. Description: mins-12--minl-10
Started at: 1732858467.993502
x - Avg MSE=360.7314, Best MSE=324.7620 at index 2
y - Avg MSE=397.6804, Best MSE=309.2426 at index 1
w - Avg MSE=1221.0095, Best MSE=1083.7336 at index 2
h - Avg MSE=1124.3736, Best MSE=985.6330 at index 0
Ended at 1732858539.8387918


Training model RandomForestRegressor. Description: mins-12--minl-12
Started at: 1732858539.838901
x - Avg MSE=396.6108, Best MSE=370.1791 at index 1
y - Avg MSE=423.5424, Best MSE=344.7904 at index 1
w - Avg MSE=1406.9043, Best MSE=1255.5046 at index 0
h - Avg MSE=1316.6607, Best MSE=1147.2850 at index 0
Ended at 1732858606.793222


Training model RandomForestRegressor. Description: mins-12--minl-14
Started at: 1732858606.7932742
x - Avg MSE=400.5662, Best MSE=364.5680 at index 2
y - Avg MSE=457.6092, Best MSE=360.0932 at index 1
w - Avg MSE=1432.8053, Best MSE=1209.0362 at index 0
h - Avg MSE=1351.3050, Best MSE=1192.4676 at index 0
Ended at 1732858670.543082
```

In [129…
```python
with open(os.path.join(model_path, "RandomForestRegressor_mins-minl_173285867054318.sav"), "rb") as rfr_grid1_e
    rfr_grid1_exp_loaded = pickle.load(rfr_grid1_exp_f)
```

In [130…
```python
rfr_grid1_exp_loaded
```

Out[130…
```
[{'mins': 4,
  'minl': 6,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(326.06745338827005),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=4,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(327.70875259964413),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=4,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1074.9533748950205),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=4,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1065.0378871162209),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=4,
                          n_estimators=30)}}},
 {'mins': 4,
  'minl': 8,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(338.88161377775975),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=4,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(363.20795886109227),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=4,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1130.0577615012533),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=4,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(1093.0108751455793),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=4,
                          n_estimators=30)}}},
 {'mins': 4,
  'minl': 10,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(362.3157194593953),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=4,
```

```
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(390.4444703648881),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=4,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1202.007618379573),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=4,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1182.0631734605765),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=4,
                         n_estimators=30)}}},
{'mins': 4,
 'minl': 12,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(387.20468317601643),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=4,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(457.0623033138475),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=4,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1349.5902237054179),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=4,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1321.2614176327272),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=4,
                         n_estimators=30)}}},
{'mins': 4,
 'minl': 14,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(414.6892158533924),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=4,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(448.28271511739257),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=4,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1494.3688645655095),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=4,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1376.1245549369203),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=4,
                         n_estimators=30)}}},
{'mins': 6,
 'minl': 6,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(308.1539217794009),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=6,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(333.9431873164692),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=6,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(999.967472812004),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=6,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(1008.4652080089289),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=6,
                         n_estimators=30)}}},
{'mins': 6,
```

```
 'minl': 8,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(335.15048527944515),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=6,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(357.57333379527955),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=6,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1131.7023522789088),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=6,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1110.7058175888708),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=6,
                         n_estimators=30)}}},
{'mins': 6,
 'minl': 10,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(384.0030157165013),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=6,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(399.1748523740218),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=6,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1229.0134895620104),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=6,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1225.1177214192164),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=6,
                         n_estimators=30)}}},
{'mins': 6,
 'minl': 12,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(382.8800231967921),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=6,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(432.51449560196966),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=6,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1313.8871474541636),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=6,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(1277.195780774294),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=6,
                         n_estimators=30)}}},
{'mins': 6,
 'minl': 14,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(428.1226832540919),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=6,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(452.19290342281937),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=6,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1428.0822107367776),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=6,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
```

```
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1358.7428695821561),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=6,
                          n_estimators=30)}}},
 {'mins': 8,
  'minl': 6,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(306.85158877880417),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(327.1678643860244),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1046.273989049578),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(960.6163423888355),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                          n_estimators=30)}}},
 {'mins': 8,
  'minl': 8,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(360.60547078092463),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=8,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(366.3724887480476),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=8,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(1169.0414180583687),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=8,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(1104.964060406972),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=8,
                          n_estimators=30)}}},
 {'mins': 8,
  'minl': 10,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(355.6135723175048),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=8,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(394.22062920870053),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=8,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1264.0395168794987),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=8,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1225.3672064631035),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=8,
                          n_estimators=30)}}},
 {'mins': 8,
  'minl': 12,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(389.969535398523),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=8,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(430.539461402906),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=8,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
```

```
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1366.1019401741187),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=8,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1260.9886560840166),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=8,
                          n_estimators=30)}}},
 {'mins': 8,
  'minl': 14,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(430.4666414007852),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=8,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(466.1908778836112),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=8,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1477.8264851209715),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=8,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1375.955870948242),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=8,
                          n_estimators=30)}}},
 {'mins': 10,
  'minl': 6,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(299.8076906355798),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=10,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(340.31863106658733),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=10,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(1105.3671235851016),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=10,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1047.4830046121215),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=10,
                          n_estimators=30)}}},
 {'mins': 10,
  'minl': 8,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(331.7603802139086),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=10,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(368.88656847103715),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=10,
                          n_estimators=30)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1190.1596522473353),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=10,
                          n_estimators=30)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1051.5488568602568),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=10,
                          n_estimators=30)}}},
 {'mins': 10,
  'minl': 10,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(355.70161495512485),
    'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=10,
                          n_estimators=30)},
   'y': {'Best MSE': inf,
```

```
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(375.95156000364915),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=10,
                           n_estimators=30)},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(2),
     'Avg MSE': np.float64(1236.8030800389367),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=10,
                           n_estimators=30)},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1217.0728006534935),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=10,
                           n_estimators=30)}}},
 {'mins': 10,
  'minl': 12,
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(2),
     'Avg MSE': np.float64(412.55929922686414),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=10,
                           n_estimators=30)},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(415.56668552456125),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=10,
                           n_estimators=30)},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1354.440820983136),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=10,
                           n_estimators=30)},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1227.357001104162),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=10,
                           n_estimators=30)}}},
 {'mins': 10,
  'minl': 14,
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(410.95453958688796),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=10,
                           n_estimators=30)},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(456.50574691078555),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=10,
                           n_estimators=30)},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1452.3752861979983),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=10,
                           n_estimators=30)},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1384.1885892652538),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=10,
                           n_estimators=30)}}},
 {'mins': 12,
  'minl': 6,
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(326.47910638442926),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=12,
                           n_estimators=30)},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(322.24707451908006),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=12,
                           n_estimators=30)},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1024.0749827613665),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=12,
                           n_estimators=30)},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1012.155106572413),
     'model': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=12,
                           n_estimators=30)}}},
 {'mins': 12,
  'minl': 8,
  'exp': {'x': {'Best MSE': inf,
```

```
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(348.10094378489345),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=12,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(372.88241966327223),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=12,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1175.6589965587987),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=12,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1065.9765569131284),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=8, min_samples_split=12,
                         n_estimators=30)}}},
{'mins': 12,
 'minl': 10,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(360.73136992040077),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=12,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(397.6803946134448),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=12,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(1221.0094692178918),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=12,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1124.373624480486),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=10, min_samples_split=12,
                         n_estimators=30)}}},
{'mins': 12,
 'minl': 12,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(396.6107753042181),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=12,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(423.54240050825547),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=12,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1406.9043253441744),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=12,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1316.6607088977933),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=12, min_samples_split=12,
                         n_estimators=30)}}},
{'mins': 12,
 'minl': 14,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(400.5661553197953),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=12,
                         n_estimators=30)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(457.6091756036264),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=12,
                         n_estimators=30)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1432.805308713826),
   'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=12,
                         n_estimators=30)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1351.3050286819214),
```

```
        'model': RandomForestRegressor(max_depth=19, min_samples_leaf=14, min_samples_split=12,
                                       n_estimators=30)}}}]
```

In [131…
```
Y_test = get_labels(data_type="Test")
X_test = get_images(data_type="Test", image_names=Y_test[:,0])
rfr_grid1_results = test(exp_list=rfr_grid1_exp_loaded, Y_test=Y_test, X_test=X_test)
```

In [132…
```
rfr_grid1_results
```

Out[132…
```
[{'mins': 4, 'minl': 6, 'weighted_avg_mse': np.float64(1030.0319577545959)},
 {'mins': 4, 'minl': 8, 'weighted_avg_mse': np.float64(1077.343572941986)},
 {'mins': 4, 'minl': 10, 'weighted_avg_mse': np.float64(965.7390259597503)},
 {'mins': 4, 'minl': 12, 'weighted_avg_mse': np.float64(1001.5366133034171)},
 {'mins': 4, 'minl': 14, 'weighted_avg_mse': np.float64(1104.5554561859)},
 {'mins': 6, 'minl': 6, 'weighted_avg_mse': np.float64(1059.7983548651566)},
 {'mins': 6, 'minl': 8, 'weighted_avg_mse': np.float64(1042.7518063871962)},
 {'mins': 6, 'minl': 10, 'weighted_avg_mse': np.float64(968.3559043781738)},
 {'mins': 6, 'minl': 12, 'weighted_avg_mse': np.float64(1010.3198934567595)},
 {'mins': 6, 'minl': 14, 'weighted_avg_mse': np.float64(1069.8815027499604)},
 {'mins': 8, 'minl': 6, 'weighted_avg_mse': np.float64(923.2098763833342)},
 {'mins': 8, 'minl': 8, 'weighted_avg_mse': np.float64(1017.9480888058059)},
 {'mins': 8, 'minl': 10, 'weighted_avg_mse': np.float64(1001.0090268207798)},
 {'mins': 8, 'minl': 12, 'weighted_avg_mse': np.float64(994.3513751375535)},
 {'mins': 8, 'minl': 14, 'weighted_avg_mse': np.float64(975.0766309208572)},
 {'mins': 10, 'minl': 6, 'weighted_avg_mse': np.float64(1013.3827100186105)},
 {'mins': 10, 'minl': 8, 'weighted_avg_mse': np.float64(1025.8023477009074)},
 {'mins': 10, 'minl': 10, 'weighted_avg_mse': np.float64(1052.0913655828344)},
 {'mins': 10, 'minl': 12, 'weighted_avg_mse': np.float64(1030.5990940395427)},
 {'mins': 10, 'minl': 14, 'weighted_avg_mse': np.float64(1025.600643111045)},
 {'mins': 12, 'minl': 6, 'weighted_avg_mse': np.float64(987.2579517063418)},
 {'mins': 12, 'minl': 8, 'weighted_avg_mse': np.float64(995.0606239828946)},
 {'mins': 12, 'minl': 10, 'weighted_avg_mse': np.float64(1096.0163546806302)},
 {'mins': 12, 'minl': 12, 'weighted_avg_mse': np.float64(969.2809944903406)},
 {'mins': 12, 'minl': 14, 'weighted_avg_mse': np.float64(981.2242151196722)}]
```

3.1.3 Store Best RandomForestRegressor Model

In [133…
```
rfr_grid1_mean = np.mean([res["weighted_avg_mse"] for res in rfr_grid1_results])
rfr_grid1_best = min(rfr_grid1_results, key=lambda x: x["weighted_avg_mse"])
rfr_grid1_best_index = rfr_grid1_results.index(rfr_grid1_best)
print(f"Best:{rfr_grid1_best}\nDiff:{rfr_grid1_best["weighted_avg_mse"]-rfr_grid1_mean}\nAt:{rfr_grid1_best_ind
```

```
Best:{'mins': 8, 'minl': 6, 'weighted_avg_mse': np.float64(923.2098763833342)}
Diff:-93.51913907602784
At:10
```

In [134…
```
time_str = str(time.time()).replace(".", "")
best_model_rfr_exp = rfr_grid1_exp_loaded[rfr_grid1_best_index] # Experiment object where the best model is sto
best_model_rfr = {
    "x":best_model_rfr_exp["exp"]["x"]["model"],
    "y":best_model_rfr_exp["exp"]["y"]["model"],
    "w":best_model_rfr_exp["exp"]["w"]["model"],
    "h":best_model_rfr_exp["exp"]["h"]["model"],
}
pickle.dump(best_model_rfr,
           open(os.path.join(model_path, f"best_model_rfr_{time_str}.sav"),"wb"))
```

3.2 Grid Search: Support Vector Regressor

Hyperparameters for Support Vector Regressor:

- `krnl` : `kernel` , the kernel of the non-linear SVM.
- `C` : `C` , trade of factor between margin and wrong samples.
- `epsl` : `epsilon` , margin of tolerance around predicted values.
- `gamm` : `gamma` , defines how far the influence of a single training example reaches.

In [200…
```
svr_krnl = ["linear", "poly", "rbf", "sigmoid"]
svr_C = [1e-2, 1e-1, 1, 10, 100]
svr_epsl = [1e-2, 5e-2, 1e-1, 5e-1, 1]
svr_gamm = ["scale", "auto", 1e-2, 1e-1, 1]
svr_grid2 = product(svr_krnl, svr_C)
svr_grid3 = product(svr_epsl, svr_gamm)
```

3.2.1 Grid Search 2: Kernel and C

In [79]:
```
svr_grid2_exp = grid_search(ModelClass=SVR,
                            hyper_params=svr_grid2,
                            hyper_param_names=["krnl", "C"],
```

```
                             kwarg_names=["kernel", "C"])
```

```
Training model SVR. Description: krnl-linear--C-0.01
Started at: 1732885883.7892659
x - Avg MSE=154.3559, Best MSE=144.1069 at index 0
y - Avg MSE=184.7328, Best MSE=175.3178 at index 0
w - Avg MSE=346.2883, Best MSE=312.1882 at index 2
h - Avg MSE=497.9693, Best MSE=429.7424 at index 2
Ended at 1732885887.438577


Training model SVR. Description: krnl-linear--C-0.1
Started at: 1732885887.438615
x - Avg MSE=154.3559, Best MSE=144.1069 at index 0
y - Avg MSE=184.7328, Best MSE=175.3178 at index 0
w - Avg MSE=346.2883, Best MSE=312.1882 at index 2
h - Avg MSE=497.9693, Best MSE=429.7424 at index 2
Ended at 1732885891.076158


Training model SVR. Description: krnl-linear--C-1
Started at: 1732885891.076183
x - Avg MSE=154.3559, Best MSE=144.1069 at index 0
y - Avg MSE=184.7328, Best MSE=175.3178 at index 0
w - Avg MSE=346.2883, Best MSE=312.1882 at index 2
h - Avg MSE=497.9693, Best MSE=429.7424 at index 2
Ended at 1732885894.703306


Training model SVR. Description: krnl-linear--C-10
Started at: 1732885894.703443
x - Avg MSE=154.3559, Best MSE=144.1069 at index 0
y - Avg MSE=184.7328, Best MSE=175.3178 at index 0
w - Avg MSE=346.2883, Best MSE=312.1882 at index 2
h - Avg MSE=497.9693, Best MSE=429.7424 at index 2
Ended at 1732885898.306389


Training model SVR. Description: krnl-linear--C-100
Started at: 1732885898.306508
x - Avg MSE=154.3559, Best MSE=144.1069 at index 0
y - Avg MSE=184.7328, Best MSE=175.3178 at index 0
w - Avg MSE=346.2883, Best MSE=312.1882 at index 2
h - Avg MSE=497.9693, Best MSE=429.7424 at index 2
Ended at 1732885902.000722


Training model SVR. Description: krnl-poly--C-0.01
Started at: 1732885902.000871
x - Avg MSE=900.1368, Best MSE=813.8476 at index 2
y - Avg MSE=1419.7872, Best MSE=1350.0775 at index 0
w - Avg MSE=5117.6814, Best MSE=4554.2961 at index 0
h - Avg MSE=4838.2068, Best MSE=4295.5821 at index 0
Ended at 1732885904.000462


Training model SVR. Description: krnl-poly--C-0.1
Started at: 1732885904.0005078
x - Avg MSE=846.6235, Best MSE=750.9788 at index 2
y - Avg MSE=1313.2814, Best MSE=1227.4244 at index 0
w - Avg MSE=4494.0145, Best MSE=4090.9379 at index 0
h - Avg MSE=4147.3053, Best MSE=3754.5803 at index 0
Ended at 1732885905.978851


Training model SVR. Description: krnl-poly--C-1
Started at: 1732885905.978983
x - Avg MSE=672.0959, Best MSE=576.5439 at index 2
y - Avg MSE=891.3330, Best MSE=832.1370 at index 0
w - Avg MSE=2574.9339, Best MSE=2275.1830 at index 0
h - Avg MSE=2369.4762, Best MSE=2148.0367 at index 0
Ended at 1732885907.968416


Training model SVR. Description: krnl-poly--C-10
Started at: 1732885907.9684658
x - Avg MSE=382.6956, Best MSE=298.8897 at index 1
y - Avg MSE=452.1923, Best MSE=422.8060 at index 0
w - Avg MSE=1194.5595, Best MSE=1051.9923 at index 1
h - Avg MSE=1095.8407, Best MSE=1004.8301 at index 1
Ended at 1732885910.021605


Training model SVR. Description: krnl-poly--C-100
```

```
Started at: 1732885910.0217311
x - Avg MSE=224.8615, Best MSE=205.8520 at index 1
y - Avg MSE=302.7942, Best MSE=282.8373 at index 0
w - Avg MSE=679.7936, Best MSE=521.9794 at index 1
h - Avg MSE=725.0279, Best MSE=702.1854 at index 0
Ended at 1732885912.399449


Training model SVR. Description: krnl-rbf--C-0.01
Started at: 1732885912.399582
x - Avg MSE=904.2718, Best MSE=824.0082 at index 2
y - Avg MSE=1433.5681, Best MSE=1361.5384 at index 0
w - Avg MSE=5259.9276, Best MSE=4651.4614 at index 0
h - Avg MSE=4997.3530, Best MSE=4384.2343 at index 0
Ended at 1732885914.990347


Training model SVR. Description: krnl-rbf--C-0.1
Started at: 1732885914.9904761
x - Avg MSE=903.7715, Best MSE=822.1107 at index 2
y - Avg MSE=1432.3463, Best MSE=1363.5609 at index 0
w - Avg MSE=5235.8638, Best MSE=4628.7088 at index 0
h - Avg MSE=4969.8623, Best MSE=4356.8919 at index 0
Ended at 1732885917.60111


Training model SVR. Description: krnl-rbf--C-1
Started at: 1732885917.601156
x - Avg MSE=891.9854, Best MSE=805.4997 at index 2
y - Avg MSE=1404.4577, Best MSE=1333.9578 at index 0
w - Avg MSE=5042.1147, Best MSE=4458.5852 at index 0
h - Avg MSE=4750.1555, Best MSE=4198.7299 at index 0
Ended at 1732885920.218326


Training model SVR. Description: krnl-rbf--C-10
Started at: 1732885920.21841
x - Avg MSE=792.0855, Best MSE=693.4426 at index 2
y - Avg MSE=1227.4440, Best MSE=1144.5136 at index 0
w - Avg MSE=3854.9410, Best MSE=3443.6127 at index 0
h - Avg MSE=3637.1657, Best MSE=3198.5658 at index 0
Ended at 1732885922.891287


Training model SVR. Description: krnl-rbf--C-100
Started at: 1732885922.891335
x - Avg MSE=526.3954, Best MSE=431.8823 at index 1
y - Avg MSE=656.4794, Best MSE=588.7548 at index 1
w - Avg MSE=1733.3432, Best MSE=1566.9456 at index 0
h - Avg MSE=1592.2043, Best MSE=1406.4131 at index 0
Ended at 1732885925.572624


Training model SVR. Description: krnl-sigmoid--C-0.01
Started at: 1732885925.572653
x - Avg MSE=904.2673, Best MSE=824.2169 at index 2
y - Avg MSE=1433.7375, Best MSE=1361.3731 at index 0
w - Avg MSE=5262.0034, Best MSE=4652.4794 at index 0
h - Avg MSE=5000.5466, Best MSE=4387.5661 at index 0
Ended at 1732885927.5948942


Training model SVR. Description: krnl-sigmoid--C-0.1
Started at: 1732885927.595084
x - Avg MSE=904.2505, Best MSE=824.2015 at index 2
y - Avg MSE=1433.7574, Best MSE=1361.4733 at index 0
w - Avg MSE=5262.1368, Best MSE=4652.5994 at index 0
h - Avg MSE=5000.7941, Best MSE=4387.8099 at index 0
Ended at 1732885929.610518


Training model SVR. Description: krnl-sigmoid--C-1
Started at: 1732885929.610676
x - Avg MSE=904.1914, Best MSE=824.0368 at index 2
y - Avg MSE=1433.9540, Best MSE=1362.5043 at index 0
w - Avg MSE=5263.4794, Best MSE=4653.8055 at index 0
h - Avg MSE=5003.2890, Best MSE=4390.2717 at index 0
Ended at 1732885931.623591


Training model SVR. Description: krnl-sigmoid--C-10
Started at: 1732885931.62372
x - Avg MSE=905.1359, Best MSE=823.0558 at index 2
```

```
y - Avg MSE=1433.9428, Best MSE=1370.1577 at index 0
w - Avg MSE=5278.8467, Best MSE=4673.5921 at index 0
h - Avg MSE=5027.9287, Best MSE=4417.2456 at index 0
Ended at 1732885933.6401038


Training model SVR. Description: krnl-sigmoid--C-100
Started at: 1732885933.640248
x - Avg MSE=1574.0886, Best MSE=1385.6201 at index 2
y - Avg MSE=1910.0432, Best MSE=1836.6530 at index 0
w - Avg MSE=5803.8740, Best MSE=5160.6499 at index 0
h - Avg MSE=5609.9626, Best MSE=4734.1798 at index 0
Ended at 1732885935.665422
```

In [112…

```python
with open(os.path.join(model_path, "SVR_krnl-C_1732885935665518.sav"), "rb") as svr_grid2_exp_f:
    svr_grid2_exp_loaded = pickle.load(svr_grid2_exp_f)
```

In [113…

```python
svr_grid2_exp_loaded
```

Out[113…

```
[{'krnl': 'linear',
  'C': 0.01,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(154.35587127921224),
    'model': SVR(C=0.01, kernel='linear')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(184.73284997785478),
    'model': SVR(C=0.01, kernel='linear')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(346.2882899757603),
    'model': SVR(C=0.01, kernel='linear')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(497.9693200481542),
    'model': SVR(C=0.01, kernel='linear')}}},
 {'krnl': 'linear',
  'C': 0.1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(154.35587127921224),
    'model': SVR(C=0.1, kernel='linear')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(184.73284997785478),
    'model': SVR(C=0.1, kernel='linear')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(346.2882899757603),
    'model': SVR(C=0.1, kernel='linear')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(497.9693200481542),
    'model': SVR(C=0.1, kernel='linear')}}},
 {'krnl': 'linear',
  'C': 1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(154.35587127921224),
    'model': SVR(C=1, kernel='linear')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(184.73284997785478),
    'model': SVR(C=1, kernel='linear')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(346.2882899757603),
    'model': SVR(C=1, kernel='linear')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(497.9693200481542),
    'model': SVR(C=1, kernel='linear')}}},
 {'krnl': 'linear',
  'C': 10,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(154.35587127921224),
    'model': SVR(C=10, kernel='linear')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
```

```
   'Avg MSE': np.float64(184.73284997785478),
   'model': SVR(C=10, kernel='linear')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(346.2882899757603),
   'model': SVR(C=10, kernel='linear')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(497.9693200481542),
   'model': SVR(C=10, kernel='linear')}}},
{'krnl': 'linear',
 'C': 100,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(154.35587127921224),
   'model': SVR(C=100, kernel='linear')},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(184.73284997785478),
   'model': SVR(C=100, kernel='linear')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(346.2882899757603),
   'model': SVR(C=100, kernel='linear')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(497.9693200481542),
   'model': SVR(C=100, kernel='linear')}}},
{'krnl': 'poly',
 'C': 0.01,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(900.1368156603307),
   'model': SVR(C=0.01, kernel='poly')},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1419.787205367895),
   'model': SVR(C=0.01, kernel='poly')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5117.681440540192),
   'model': SVR(C=0.01, kernel='poly')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(4838.206811653575),
   'model': SVR(C=0.01, kernel='poly')}}},
{'krnl': 'poly',
 'C': 0.1,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(846.6235060263831),
   'model': SVR(C=0.1, kernel='poly')},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1313.2813970845336),
   'model': SVR(C=0.1, kernel='poly')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(4494.0145253218325),
   'model': SVR(C=0.1, kernel='poly')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(4147.305303759066),
   'model': SVR(C=0.1, kernel='poly')}}},
{'krnl': 'poly',
 'C': 1,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(672.0959445262671),
   'model': SVR(C=1, kernel='poly')},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(891.332952915552),
   'model': SVR(C=1, kernel='poly')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(2574.9339269850657),
   'model': SVR(C=1, kernel='poly')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(2369.4762060150497),
   'model': SVR(C=1, kernel='poly')}}},
{'krnl': 'poly',
```

```
  'C': 10,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(382.6956046097661),
    'model': SVR(C=10, kernel='poly')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(452.19231055440713),
    'model': SVR(C=10, kernel='poly')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(1194.5595331847817),
    'model': SVR(C=10, kernel='poly')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(1095.8406555335462),
    'model': SVR(C=10, kernel='poly')}}},
 {'krnl': 'poly',
  'C': 100,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(224.86148430646313),
    'model': SVR(C=100, kernel='poly')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(302.794196175975),
    'model': SVR(C=100, kernel='poly')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(679.7935941313541),
    'model': SVR(C=100, kernel='poly')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(725.0279465992198),
    'model': SVR(C=100, kernel='poly')}}},
 {'krnl': 'rbf',
  'C': 0.01,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(904.2718425254155),
    'model': SVR(C=0.01)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1433.5680782674317),
    'model': SVR(C=0.01)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5259.927591998764),
    'model': SVR(C=0.01)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4997.35297938784),
    'model': SVR(C=0.01)}}},
 {'krnl': 'rbf',
  'C': 0.1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(903.7714926725122),
    'model': SVR(C=0.1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1432.3463319527448),
    'model': SVR(C=0.1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5235.863828691613),
    'model': SVR(C=0.1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4969.86231547706),
    'model': SVR(C=0.1)}}},
 {'krnl': 'rbf',
  'C': 1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(891.9854055946233),
    'model': SVR(C=1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1404.457686864913),
    'model': SVR(C=1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
```

```
    'Avg MSE': np.float64(5042.11472759622),
    'model': SVR(C=1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4750.155478106952),
    'model': SVR(C=1)}}},
 {'krnl': 'rbf',
  'C': 10,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(792.0855183931862),
    'model': SVR(C=10)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1227.444029627596),
    'model': SVR(C=10)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(3854.9410393112007),
    'model': SVR(C=10)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(3637.165746440376),
    'model': SVR(C=10)}}},
 {'krnl': 'rbf',
  'C': 100,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(526.3953969484345),
    'model': SVR(C=100)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(656.4794102639777),
    'model': SVR(C=100)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1733.3431622085225),
    'model': SVR(C=100)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1592.2043411488794),
    'model': SVR(C=100)}}},
 {'krnl': 'sigmoid',
  'C': 0.01,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(904.2673266229394),
    'model': SVR(C=0.01, kernel='sigmoid')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1433.7375160522631),
    'model': SVR(C=0.01, kernel='sigmoid')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5262.003449677238),
    'model': SVR(C=0.01, kernel='sigmoid')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5000.546619275924),
    'model': SVR(C=0.01, kernel='sigmoid')}}},
 {'krnl': 'sigmoid',
  'C': 0.1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(904.2504947970228),
    'model': SVR(C=0.1, kernel='sigmoid')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1433.757391805376),
    'model': SVR(C=0.1, kernel='sigmoid')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5262.136845782562),
    'model': SVR(C=0.1, kernel='sigmoid')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5000.7940508654765),
    'model': SVR(C=0.1, kernel='sigmoid')}}},
 {'krnl': 'sigmoid',
  'C': 1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(2),
    'Avg MSE': np.float64(904.1913901775168),
```

```
   'model': SVR(C=1, kernel='sigmoid')},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1433.9539614406456),
   'model': SVR(C=1, kernel='sigmoid')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5263.479371343511),
   'model': SVR(C=1, kernel='sigmoid')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5003.288969281591),
   'model': SVR(C=1, kernel='sigmoid')}}},
{'krnl': 'sigmoid',
 'C': 10,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(905.1358732019804),
   'model': SVR(C=10, kernel='sigmoid')},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1433.942822428714),
   'model': SVR(C=10, kernel='sigmoid')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5278.846708955911),
   'model': SVR(C=10, kernel='sigmoid')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5027.928730773035),
   'model': SVR(C=10, kernel='sigmoid')}}},
{'krnl': 'sigmoid',
 'C': 100,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(2),
   'Avg MSE': np.float64(1574.0886297663667),
   'model': SVR(C=100, kernel='sigmoid')},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1910.0431564206513),
   'model': SVR(C=100, kernel='sigmoid')},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5803.873955971059),
   'model': SVR(C=100, kernel='sigmoid')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5609.962574465761),
   'model': SVR(C=100, kernel='sigmoid')}}}]
```

In [114…

```
Y_test = get_labels(data_type="Test")
X_test = get_images(data_type="Test", image_names=Y_test[:,0])
svr_grid2_results = test(exp_list=svr_grid2_exp_loaded, Y_test=Y_test, X_test=X_test)
```

In [115…

```
svr_grid2_results
```

Out[115…
```
[{'krnl': 'linear',
  'C': 0.01,
  'weighted_avg_mse': np.float64(1581.1840668728994)},
 {'krnl': 'linear',
  'C': 0.1,
  'weighted_avg_mse': np.float64(1581.1840668728994)},
 {'krnl': 'linear',
  'C': 1,
  'weighted_avg_mse': np.float64(1581.1840668728994)},
 {'krnl': 'linear',
  'C': 10,
  'weighted_avg_mse': np.float64(1581.1840668728994)},
 {'krnl': 'linear',
  'C': 100,
  'weighted_avg_mse': np.float64(1581.1840668728994)},
 {'krnl': 'poly',
  'C': 0.01,
  'weighted_avg_mse': np.float64(1880.1085308176234)},
 {'krnl': 'poly', 'C': 0.1, 'weighted_avg_mse': np.float64(1614.385705927301)},
 {'krnl': 'poly', 'C': 1, 'weighted_avg_mse': np.float64(923.7676284396246)},
 {'krnl': 'poly', 'C': 10, 'weighted_avg_mse': np.float64(892.2569922758053)},
 {'krnl': 'poly',
  'C': 100,
  'weighted_avg_mse': np.float64(1227.5168107888653)},
 {'krnl': 'rbf',
  'C': 0.01,
  'weighted_avg_mse': np.float64(1939.4852304151748)},
 {'krnl': 'rbf', 'C': 0.1, 'weighted_avg_mse': np.float64(1930.8324008354684)},
 {'krnl': 'rbf', 'C': 1, 'weighted_avg_mse': np.float64(1845.9187159982262)},
 {'krnl': 'rbf', 'C': 10, 'weighted_avg_mse': np.float64(1398.5448203731798)},
 {'krnl': 'rbf', 'C': 100, 'weighted_avg_mse': np.float64(788.4857448259537)},
 {'krnl': 'sigmoid',
  'C': 0.01,
  'weighted_avg_mse': np.float64(1940.3281567088688)},
 {'krnl': 'sigmoid',
  'C': 0.1,
  'weighted_avg_mse': np.float64(1940.3444406818187)},
 {'krnl': 'sigmoid',
  'C': 1,
  'weighted_avg_mse': np.float64(1940.4732162311595)},
 {'krnl': 'sigmoid',
  'C': 10,
  'weighted_avg_mse': np.float64(1942.6019290575855)},
 {'krnl': 'sigmoid',
  'C': 100,
  'weighted_avg_mse': np.float64(2446.5445497979276)}]
```

In [116…
```
svr_grid2_mean = np.mean([res["weighted_avg_mse"] for res in svr_grid2_results])
svr_grid2_best = min(svr_grid2_results, key=lambda x: x["weighted_avg_mse"])
print(f"Best:{svr_grid2_best}\nDiff=:{svr_grid2_best["weighted_avg_mse"]-svr_grid2_mean}")
```

```
Best:{'krnl': 'rbf', 'C': 100, 'weighted_avg_mse': np.float64(788.4857448259537)}
Diff=:-839.3900155510004
```

3.2.2 Grid Search 3: Epsilon and Gamma

In [90]:
```
svr_grid3_exp = grid_search(ModelClass=SVR,
                            hyper_params=svr_grid3,
                            hyper_param_names=["epsl", "gamm"],
                            kwarg_names=["epsilon", "gamma"],
                            # Settled Parameters
                            kernel="rbf",
                            C=100)
```

```
Training model SVR. Description: epsl-0.01--gamm-scale
Started at: 1732886577.4301
x - Avg MSE=526.5641, Best MSE=432.0645 at index 1
y - Avg MSE=656.2642, Best MSE=588.4727 at index 1
w - Avg MSE=1733.6229, Best MSE=1567.1903 at index 0
h - Avg MSE=1592.6447, Best MSE=1406.3219 at index 0
Ended at 1732886580.112839


Training model SVR. Description: epsl-0.01--gamm-auto
Started at: 1732886580.112881
x - Avg MSE=891.0212, Best MSE=826.4564 at index 1
y - Avg MSE=1413.6084, Best MSE=1354.1260 at index 0
w - Avg MSE=5205.9253, Best MSE=4625.3975 at index 0
h - Avg MSE=4934.5779, Best MSE=4342.2064 at index 0
Ended at 1732886582.783846


Training model SVR. Description: epsl-0.01--gamm-0.01
```

```
Started at: 1732886582.7838762
x - Avg MSE=891.0212, Best MSE=826.4564 at index 1
y - Avg MSE=1413.6084, Best MSE=1354.1260 at index 0
w - Avg MSE=5205.9253, Best MSE=4625.3975 at index 0
h - Avg MSE=4934.5779, Best MSE=4342.2064 at index 0
Ended at 1732886585.437207


Training model SVR. Description: epsl-0.01--gamm-0.1
Started at: 1732886585.437232
x - Avg MSE=891.0212, Best MSE=826.4564 at index 1
y - Avg MSE=1413.6084, Best MSE=1354.1260 at index 0
w - Avg MSE=5205.9253, Best MSE=4625.3975 at index 0
h - Avg MSE=4934.5779, Best MSE=4342.2064 at index 0
Ended at 1732886588.060579


Training model SVR. Description: epsl-0.01--gamm-1
Started at: 1732886588.0607219
x - Avg MSE=891.0212, Best MSE=826.4564 at index 1
y - Avg MSE=1413.6084, Best MSE=1354.1260 at index 0
w - Avg MSE=5205.9253, Best MSE=4625.3975 at index 0
h - Avg MSE=4934.5779, Best MSE=4342.2064 at index 0
Ended at 1732886590.6861901


Training model SVR. Description: epsl-0.05--gamm-scale
Started at: 1732886590.686235
x - Avg MSE=526.4832, Best MSE=431.9815 at index 1
y - Avg MSE=656.3718, Best MSE=588.6035 at index 1
w - Avg MSE=1733.4921, Best MSE=1567.1005 at index 0
h - Avg MSE=1592.4511, Best MSE=1406.3534 at index 0
Ended at 1732886593.332267


Training model SVR. Description: epsl-0.05--gamm-auto
Started at: 1732886593.332313
x - Avg MSE=891.0221, Best MSE=826.4571 at index 1
y - Avg MSE=1413.6074, Best MSE=1354.1291 at index 0
w - Avg MSE=5205.8999, Best MSE=4625.3850 at index 0
h - Avg MSE=4934.5685, Best MSE=4342.2068 at index 0
Ended at 1732886595.971176


Training model SVR. Description: epsl-0.05--gamm-0.01
Started at: 1732886595.9712949
x - Avg MSE=891.0221, Best MSE=826.4571 at index 1
y - Avg MSE=1413.6074, Best MSE=1354.1291 at index 0
w - Avg MSE=5205.8999, Best MSE=4625.3850 at index 0
h - Avg MSE=4934.5685, Best MSE=4342.2068 at index 0
Ended at 1732886598.590092


Training model SVR. Description: epsl-0.05--gamm-0.1
Started at: 1732886598.590254
x - Avg MSE=891.0221, Best MSE=826.4571 at index 1
y - Avg MSE=1413.6074, Best MSE=1354.1291 at index 0
w - Avg MSE=5205.8999, Best MSE=4625.3850 at index 0
h - Avg MSE=4934.5685, Best MSE=4342.2068 at index 0
Ended at 1732886601.218598


Training model SVR. Description: epsl-0.05--gamm-1
Started at: 1732886601.218716
x - Avg MSE=891.0221, Best MSE=826.4571 at index 1
y - Avg MSE=1413.6074, Best MSE=1354.1291 at index 0
w - Avg MSE=5205.8999, Best MSE=4625.3850 at index 0
h - Avg MSE=4934.5685, Best MSE=4342.2068 at index 0
Ended at 1732886603.845707


Training model SVR. Description: epsl-0.1--gamm-scale
Started at: 1732886603.845866
x - Avg MSE=526.3954, Best MSE=431.8823 at index 1
y - Avg MSE=656.4794, Best MSE=588.7548 at index 1
w - Avg MSE=1733.3432, Best MSE=1566.9456 at index 0
h - Avg MSE=1592.2043, Best MSE=1406.4131 at index 0
Ended at 1732886606.502934


Training model SVR. Description: epsl-0.1--gamm-auto
Started at: 1732886606.50298
x - Avg MSE=891.0234, Best MSE=826.4580 at index 1
```

```
y - Avg MSE=1413.6063, Best MSE=1354.1330 at index 0
w - Avg MSE=5205.8682, Best MSE=4625.3695 at index 0
h - Avg MSE=4934.5569, Best MSE=4342.2073 at index 0
Ended at 1732886609.1737878


Training model SVR. Description: epsl-0.1--gamm-0.01
Started at: 1732886609.173814
x - Avg MSE=891.0234, Best MSE=826.4580 at index 1
y - Avg MSE=1413.6063, Best MSE=1354.1330 at index 0
w - Avg MSE=5205.8682, Best MSE=4625.3695 at index 0
h - Avg MSE=4934.5569, Best MSE=4342.2073 at index 0
Ended at 1732886611.790575


Training model SVR. Description: epsl-0.1--gamm-0.1
Started at: 1732886611.790601
x - Avg MSE=891.0234, Best MSE=826.4580 at index 1
y - Avg MSE=1413.6063, Best MSE=1354.1330 at index 0
w - Avg MSE=5205.8682, Best MSE=4625.3695 at index 0
h - Avg MSE=4934.5569, Best MSE=4342.2073 at index 0
Ended at 1732886614.487093


Training model SVR. Description: epsl-0.1--gamm-1
Started at: 1732886614.487155
x - Avg MSE=891.0234, Best MSE=826.4580 at index 1
y - Avg MSE=1413.6063, Best MSE=1354.1330 at index 0
w - Avg MSE=5205.8682, Best MSE=4625.3695 at index 0
h - Avg MSE=4934.5569, Best MSE=4342.2073 at index 0
Ended at 1732886617.170048


Training model SVR. Description: epsl-0.5--gamm-scale
Started at: 1732886617.1701
x - Avg MSE=525.6412, Best MSE=431.2557 at index 1
y - Avg MSE=657.1046, Best MSE=589.2915 at index 1
w - Avg MSE=1732.0910, Best MSE=1565.7336 at index 0
h - Avg MSE=1590.7126, Best MSE=1406.7905 at index 0
Ended at 1732886619.776503


Training model SVR. Description: epsl-0.5--gamm-auto
Started at: 1732886619.77667
x - Avg MSE=891.0323, Best MSE=826.4681 at index 1
y - Avg MSE=1413.5966, Best MSE=1354.1625 at index 0
w - Avg MSE=5205.6181, Best MSE=4625.2474 at index 0
h - Avg MSE=4934.4645, Best MSE=4342.2135 at index 0
Ended at 1732886622.397131


Training model SVR. Description: epsl-0.5--gamm-0.01
Started at: 1732886622.397336
x - Avg MSE=891.0323, Best MSE=826.4681 at index 1
y - Avg MSE=1413.5966, Best MSE=1354.1625 at index 0
w - Avg MSE=5205.6181, Best MSE=4625.2474 at index 0
h - Avg MSE=4934.4645, Best MSE=4342.2135 at index 0
Ended at 1732886625.0026062


Training model SVR. Description: epsl-0.5--gamm-0.1
Started at: 1732886625.00277
x - Avg MSE=891.0323, Best MSE=826.4681 at index 1
y - Avg MSE=1413.5966, Best MSE=1354.1625 at index 0
w - Avg MSE=5205.6181, Best MSE=4625.2474 at index 0
h - Avg MSE=4934.4645, Best MSE=4342.2135 at index 0
Ended at 1732886627.716536


Training model SVR. Description: epsl-0.5--gamm-1
Started at: 1732886627.716565
x - Avg MSE=891.0323, Best MSE=826.4681 at index 1
y - Avg MSE=1413.5966, Best MSE=1354.1625 at index 0
w - Avg MSE=5205.6181, Best MSE=4625.2474 at index 0
h - Avg MSE=4934.4645, Best MSE=4342.2135 at index 0
Ended at 1732886630.433105


Training model SVR. Description: epsl-1--gamm-scale
Started at: 1732886630.4332879
x - Avg MSE=524.4596, Best MSE=431.1472 at index 1
y - Avg MSE=657.8476, Best MSE=589.2239 at index 1
w - Avg MSE=1730.8319, Best MSE=1564.6114 at index 0
```

```
h - Avg MSE=1588.6821, Best MSE=1406.6098 at index 0
Ended at 1732886633.014171


Training model SVR. Description: epsl-1--gamm-auto
Started at: 1732886633.014391
x - Avg MSE=891.0554, Best MSE=826.4908 at index 1
y - Avg MSE=1413.5915, Best MSE=1354.2072 at index 0
w - Avg MSE=5205.3396, Best MSE=4625.0910 at index 0
h - Avg MSE=4934.3544, Best MSE=4342.2249 at index 0
Ended at 1732886635.614023


Training model SVR. Description: epsl-1--gamm-0.01
Started at: 1732886635.61407
x - Avg MSE=891.0554, Best MSE=826.4908 at index 1
y - Avg MSE=1413.5915, Best MSE=1354.2072 at index 0
w - Avg MSE=5205.3396, Best MSE=4625.0910 at index 0
h - Avg MSE=4934.3544, Best MSE=4342.2249 at index 0
Ended at 1732886638.2309299


Training model SVR. Description: epsl-1--gamm-0.1
Started at: 1732886638.230983
x - Avg MSE=891.0554, Best MSE=826.4908 at index 1
y - Avg MSE=1413.5915, Best MSE=1354.2072 at index 0
w - Avg MSE=5205.3396, Best MSE=4625.0910 at index 0
h - Avg MSE=4934.3544, Best MSE=4342.2249 at index 0
Ended at 1732886640.824859


Training model SVR. Description: epsl-1--gamm-1
Started at: 1732886640.824943
x - Avg MSE=891.0554, Best MSE=826.4908 at index 1
y - Avg MSE=1413.5915, Best MSE=1354.2072 at index 0
w - Avg MSE=5205.3396, Best MSE=4625.0910 at index 0
h - Avg MSE=4934.3544, Best MSE=4342.2249 at index 0
Ended at 1732886643.413711
```

In [10]:
```python
with open(os.path.join(model_path, "SVR_epsl-gamm_1732886643413752.sav"), "rb") as svr_grid3_exp_f:
    svr_grid3_exp_loaded = pickle.load(svr_grid3_exp_f)
```

In [11]:
```python
svr_grid3_exp_loaded
```

Out[11]:
```
[{'epsl': 0.01,
  'gamm': 'scale',
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(526.5640500306386),
    'model': SVR(C=100, epsilon=0.01)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(656.264165902226),
    'model': SVR(C=100, epsilon=0.01)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1733.6228936698847),
    'model': SVR(C=100, epsilon=0.01)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1592.6446730060773),
    'model': SVR(C=100, epsilon=0.01)}}},
 {'epsl': 0.01,
  'gamm': 'auto',
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0211827316698),
    'model': SVR(C=100, epsilon=0.01, gamma='auto')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6083942731318),
    'model': SVR(C=100, epsilon=0.01, gamma='auto')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.925279647147),
    'model': SVR(C=100, epsilon=0.01, gamma='auto')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.577867353456),
    'model': SVR(C=100, epsilon=0.01, gamma='auto')}}},
 {'epsl': 0.01,
```

```
  'gamm': 0.01,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0211827316698),
    'model': SVR(C=100, epsilon=0.01, gamma=0.01)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6083942731318),
    'model': SVR(C=100, epsilon=0.01, gamma=0.01)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.9252796471455),
    'model': SVR(C=100, epsilon=0.01, gamma=0.01)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.577867353457),
    'model': SVR(C=100, epsilon=0.01, gamma=0.01)}}},
 {'epsl': 0.01,
  'gamm': 0.1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0211827316698),
    'model': SVR(C=100, epsilon=0.01, gamma=0.1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6083942731318),
    'model': SVR(C=100, epsilon=0.01, gamma=0.1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.9252796471455),
    'model': SVR(C=100, epsilon=0.01, gamma=0.1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.577867353457),
    'model': SVR(C=100, epsilon=0.01, gamma=0.1)}}},
 {'epsl': 0.01,
  'gamm': 1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0211827316698),
    'model': SVR(C=100, epsilon=0.01, gamma=1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6083942731318),
    'model': SVR(C=100, epsilon=0.01, gamma=1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.9252796471455),
    'model': SVR(C=100, epsilon=0.01, gamma=1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.577867353457),
    'model': SVR(C=100, epsilon=0.01, gamma=1)}}},
 {'epsl': 0.05,
  'gamm': 'scale',
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(526.4832308451253),
    'model': SVR(C=100, epsilon=0.05)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(656.3717892054897),
    'model': SVR(C=100, epsilon=0.05)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1733.4921081272987),
    'model': SVR(C=100, epsilon=0.05)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1592.4510628310993),
    'model': SVR(C=100, epsilon=0.05)}}},
 {'epsl': 0.05,
  'gamm': 'auto',
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0221432916697),
    'model': SVR(C=100, epsilon=0.05, gamma='auto')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6074419660542),
    'model': SVR(C=100, epsilon=0.05, gamma='auto')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
```

```
   'Avg MSE': np.float64(5205.899881193801),
   'model': SVR(C=100, epsilon=0.05, gamma='auto')},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(4934.568516884226),
   'model': SVR(C=100, epsilon=0.05, gamma='auto')}}},
{'epsl': 0.05,
 'gamm': 0.01,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(891.0221432916697),
   'model': SVR(C=100, epsilon=0.05, gamma=0.01)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1413.6074419660542),
   'model': SVR(C=100, epsilon=0.05, gamma=0.01)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5205.899881193801),
   'model': SVR(C=100, epsilon=0.05, gamma=0.01)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(4934.568516884226),
   'model': SVR(C=100, epsilon=0.05, gamma=0.01)}}},
{'epsl': 0.05,
 'gamm': 0.1,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(891.0221432916697),
   'model': SVR(C=100, epsilon=0.05, gamma=0.1)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1413.6074419660542),
   'model': SVR(C=100, epsilon=0.05, gamma=0.1)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5205.899881193801),
   'model': SVR(C=100, epsilon=0.05, gamma=0.1)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(4934.568516884226),
   'model': SVR(C=100, epsilon=0.05, gamma=0.1)}}},
{'epsl': 0.05,
 'gamm': 1,
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(891.0221432916697),
   'model': SVR(C=100, epsilon=0.05, gamma=1)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1413.6074419660542),
   'model': SVR(C=100, epsilon=0.05, gamma=1)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(5205.899881193801),
   'model': SVR(C=100, epsilon=0.05, gamma=1)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(4934.568516884226),
   'model': SVR(C=100, epsilon=0.05, gamma=1)}}},
{'epsl': 0.1,
 'gamm': 'scale',
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(526.3953969484345),
   'model': SVR(C=100)},
  'y': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(656.4794102639777),
   'model': SVR(C=100)},
  'w': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1733.3431622085225),
   'model': SVR(C=100)},
  'h': {'Best MSE': inf,
   'Best Fold': np.int64(0),
   'Avg MSE': np.float64(1592.2043411488794),
   'model': SVR(C=100)}}},
{'epsl': 0.1,
 'gamm': 'auto',
 'exp': {'x': {'Best MSE': inf,
   'Best Fold': np.int64(1),
   'Avg MSE': np.float64(891.0234481666675),
```

```
    'model': SVR(C=100, gamma='auto')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6062858931764),
    'model': SVR(C=100, gamma='auto')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.868229187149),
    'model': SVR(C=100, gamma='auto')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.55686946524),
    'model': SVR(C=100, gamma='auto')}}},
 {'epsl': 0.1,
  'gamm': 0.01,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0234481666675),
    'model': SVR(C=100, gamma=0.01)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6062858931764),
    'model': SVR(C=100, gamma=0.01)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.868229187149),
    'model': SVR(C=100, gamma=0.01)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.55686946524),
    'model': SVR(C=100, gamma=0.01)}}},
 {'epsl': 0.1,
  'gamm': 0.1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0234481666675),
    'model': SVR(C=100, gamma=0.1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6062858931764),
    'model': SVR(C=100, gamma=0.1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.868229187149),
    'model': SVR(C=100, gamma=0.1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.55686946524),
    'model': SVR(C=100, gamma=0.1)}}},
 {'epsl': 0.1,
  'gamm': 1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0234481666675),
    'model': SVR(C=100, gamma=1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.6062858931764),
    'model': SVR(C=100, gamma=1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.868229187149),
    'model': SVR(C=100, gamma=1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.55686946524),
    'model': SVR(C=100, gamma=1)}}},
 {'epsl': 0.5,
  'gamm': 'scale',
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(525.6412389315909),
    'model': SVR(C=100, epsilon=0.5)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(657.104567842591),
    'model': SVR(C=100, epsilon=0.5)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1732.0910181095678),
    'model': SVR(C=100, epsilon=0.5)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
```

```
     'Avg MSE': np.float64(1590.7125693548287),
     'model': SVR(C=100, epsilon=0.5)}}},
 {'epsl': 0.5,
  'gamm': 'auto',
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(891.0323048112424),
     'model': SVR(C=100, epsilon=0.5, gamma='auto')},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1413.5965667896805),
     'model': SVR(C=100, epsilon=0.5, gamma='auto')},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(5205.618051393751),
     'model': SVR(C=100, epsilon=0.5, gamma='auto')},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(4934.464480059606),
     'model': SVR(C=100, epsilon=0.5, gamma='auto')}}},
 {'epsl': 0.5,
  'gamm': 0.01,
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(891.0323048112424),
     'model': SVR(C=100, epsilon=0.5, gamma=0.01)},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1413.5965667896805),
     'model': SVR(C=100, epsilon=0.5, gamma=0.01)},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(5205.618051393751),
     'model': SVR(C=100, epsilon=0.5, gamma=0.01)},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(4934.464480059607),
     'model': SVR(C=100, epsilon=0.5, gamma=0.01)}}},
 {'epsl': 0.5,
  'gamm': 0.1,
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(891.0323048112424),
     'model': SVR(C=100, epsilon=0.5, gamma=0.1)},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1413.5965667896805),
     'model': SVR(C=100, epsilon=0.5, gamma=0.1)},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(5205.618051393751),
     'model': SVR(C=100, epsilon=0.5, gamma=0.1)},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(4934.464480059607),
     'model': SVR(C=100, epsilon=0.5, gamma=0.1)}}},
 {'epsl': 0.5,
  'gamm': 1,
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(891.0323048112424),
     'model': SVR(C=100, epsilon=0.5, gamma=1)},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(1413.5965667896805),
     'model': SVR(C=100, epsilon=0.5, gamma=1)},
    'w': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(5205.618051393751),
     'model': SVR(C=100, epsilon=0.5, gamma=1)},
    'h': {'Best MSE': inf,
     'Best Fold': np.int64(0),
     'Avg MSE': np.float64(4934.464480059607),
     'model': SVR(C=100, epsilon=0.5, gamma=1)}}},
 {'epsl': 1,
  'gamm': 'scale',
  'exp': {'x': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(524.4596478996967),
     'model': SVR(C=100, epsilon=1)},
    'y': {'Best MSE': inf,
     'Best Fold': np.int64(1),
     'Avg MSE': np.float64(657.8475636140283),
```

```
    'model': SVR(C=100, epsilon=1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1730.8319225553926),
    'model': SVR(C=100, epsilon=1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1588.682134827092),
    'model': SVR(C=100, epsilon=1)}}},
 {'epsl': 1,
  'gamm': 'auto',
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0553536979058),
    'model': SVR(C=100, epsilon=1, gamma='auto')},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.5914935629287),
    'model': SVR(C=100, epsilon=1, gamma='auto')},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.339623170628),
    'model': SVR(C=100, epsilon=1, gamma='auto')},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.354446916778),
    'model': SVR(C=100, epsilon=1, gamma='auto')}}},
 {'epsl': 1,
  'gamm': 0.01,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0553536979058),
    'model': SVR(C=100, epsilon=1, gamma=0.01)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.5914935629287),
    'model': SVR(C=100, epsilon=1, gamma=0.01)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.339623170629),
    'model': SVR(C=100, epsilon=1, gamma=0.01)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.354446916778),
    'model': SVR(C=100, epsilon=1, gamma=0.01)}}},
 {'epsl': 1,
  'gamm': 0.1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0553536979058),
    'model': SVR(C=100, epsilon=1, gamma=0.1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.5914935629287),
    'model': SVR(C=100, epsilon=1, gamma=0.1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.339623170629),
    'model': SVR(C=100, epsilon=1, gamma=0.1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.354446916778),
    'model': SVR(C=100, epsilon=1, gamma=0.1)}}},
 {'epsl': 1,
  'gamm': 1,
  'exp': {'x': {'Best MSE': inf,
    'Best Fold': np.int64(1),
    'Avg MSE': np.float64(891.0553536979058),
    'model': SVR(C=100, epsilon=1, gamma=1)},
   'y': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(1413.5914935629287),
    'model': SVR(C=100, epsilon=1, gamma=1)},
   'w': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(5205.339623170629),
    'model': SVR(C=100, epsilon=1, gamma=1)},
   'h': {'Best MSE': inf,
    'Best Fold': np.int64(0),
    'Avg MSE': np.float64(4934.354446916778),
    'model': SVR(C=100, epsilon=1, gamma=1)}}}]
```

In [12]:
```
Y_test = get_labels(data_type="Test")
```

```
X_test = get_images(data_type="Test", image_names=Y_test[:,0])
svr_grid3_results = test(exp_list=svr_grid3_exp_loaded, Y_test=Y_test, X_test=X_test)
```

In [13]:
```
svr_grid3_results
```

Out[13]:
```
[{'epsl': 0.01,
  'gamm': 'scale',
  'weighted_avg_mse': np.float64(788.6830908613668)},
 {'epsl': 0.01,
  'gamm': 'auto',
  'weighted_avg_mse': np.float64(1856.6190083696765)},
 {'epsl': 0.01,
  'gamm': 0.01,
  'weighted_avg_mse': np.float64(1856.6190083696767)},
 {'epsl': 0.01,
  'gamm': 0.1,
  'weighted_avg_mse': np.float64(1856.6190083696767)},
 {'epsl': 0.01, 'gamm': 1, 'weighted_avg_mse': np.float64(1856.6190083696767)},
 {'epsl': 0.05,
  'gamm': 'scale',
  'weighted_avg_mse': np.float64(788.5910605860047)},
 {'epsl': 0.05,
  'gamm': 'auto',
  'weighted_avg_mse': np.float64(1856.553132874258)},
 {'epsl': 0.05,
  'gamm': 0.01,
  'weighted_avg_mse': np.float64(1856.553132874258)},
 {'epsl': 0.05,
  'gamm': 0.1,
  'weighted_avg_mse': np.float64(1856.553132874258)},
 {'epsl': 0.05, 'gamm': 1, 'weighted_avg_mse': np.float64(1856.553132874258)},
 {'epsl': 0.1,
  'gamm': 'scale',
  'weighted_avg_mse': np.float64(788.4857448259537)},
 {'epsl': 0.1,
  'gamm': 'auto',
  'weighted_avg_mse': np.float64(1856.4708598992315)},
 {'epsl': 0.1,
  'gamm': 0.01,
  'weighted_avg_mse': np.float64(1856.4708598992315)},
 {'epsl': 0.1,
  'gamm': 0.1,
  'weighted_avg_mse': np.float64(1856.4708598992315)},
 {'epsl': 0.1, 'gamm': 1, 'weighted_avg_mse': np.float64(1856.4708598992315)},
 {'epsl': 0.5,
  'gamm': 'scale',
  'weighted_avg_mse': np.float64(787.761941729204)},
 {'epsl': 0.5,
  'gamm': 'auto',
  'weighted_avg_mse': np.float64(1855.847525637413)},
 {'epsl': 0.5,
  'gamm': 0.01,
  'weighted_avg_mse': np.float64(1855.847525637413)},
 {'epsl': 0.5, 'gamm': 0.1, 'weighted_avg_mse': np.float64(1855.847525637413)},
 {'epsl': 0.5, 'gamm': 1, 'weighted_avg_mse': np.float64(1855.847525637413)},
 {'epsl': 1,
  'gamm': 'scale',
  'weighted_avg_mse': np.float64(786.886030613608)},
 {'epsl': 1,
  'gamm': 'auto',
  'weighted_avg_mse': np.float64(1855.1163311642163)},
 {'epsl': 1, 'gamm': 0.01, 'weighted_avg_mse': np.float64(1855.1163311642163)},
 {'epsl': 1, 'gamm': 0.1, 'weighted_avg_mse': np.float64(1855.1163311642163)},
 {'epsl': 1, 'gamm': 1, 'weighted_avg_mse': np.float64(1855.1163311642163)}]
```

3.2.3 Store Best SVR

In [21]:
```
svr_grid3_mean = np.mean([res["weighted_avg_mse"] for res in svr_grid3_results])
svr_grid3_best = min(svr_grid3_results, key=lambda x: x["weighted_avg_mse"])
svr_grid3_best_index = svr_grid3_results.index(svr_grid3_best)
print(f"Best:{svr_grid3_best}\nDiff=:{svr_grid3_best["weighted_avg_mse"]-svr_grid3_mean}\nAt:{svr_grid3_best_in
```

```
Best:{'epsl': 1, 'gamm': 'scale', 'weighted_avg_mse': np.float64(786.886030613608)}
Diff=:-855.627381402205
At:20
```

In [115…
```
time_str = str(time.time()).replace(".","")
best_model_svr_exp = svr_grid3_exp_loaded[svr_grid3_best_index] # Experiment object where the best model is sto
best_model_svr = {
    "x": best_model_svr_exp["exp"]["x"]["model"],
    "y": best_model_svr_exp["exp"]["y"]["model"],
    "w": best_model_svr_exp["exp"]["w"]["model"],
    "h": best_model_svr_exp["exp"]["h"]["model"],
```

```
}
pickle.dump(best_model_svr,
            open(os.path.join(model_path, f"best_model_svr_{time_str}.sav"), "wb"))
```

4. Visualization

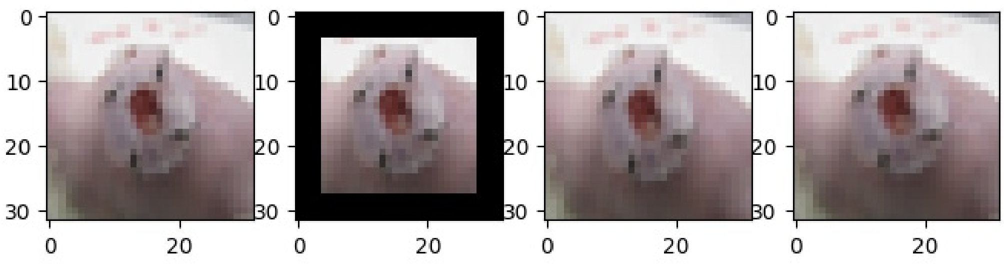
4.1 Visualize Image Augmentation

In [179…

```
def visualize_image_augmentation():
    # Data Augmentation
    # Change some useless information
    Y_ori = get_labels(data_type="Training")
    X_ori = get_images(data_type="Training", image_names=Y_ori[:, 0], flatten=False)

    # Add black edge
    Y_be = get_labels(data_type="Training")
    X_be = get_images(data_type="Training", image_names=Y_be[:, 0], augmentation=add_black_edge, w=4, flatten=Fa

    # Stretch height
    Y_sh = get_labels(data_type="Training")
    X_sh = get_images(data_type="Training", image_names=Y_sh[:, 0], augmentation=stretch, f=[1.0, 1.05], flatte
    Y_sh[:, 4] *= 1.05

    # Stretch Width
    Y_sw = get_labels(data_type="Training")
    X_sw = get_images(data_type="Training", image_names=Y_sw[:, 0], augmentation=stretch, f=[1.05, 1.0], flatte
    Y_sw[:, 3] *= 1.05

    fig, axs = plt.subplots(1,4, figsize=(8,32))
    image_samples = [X_ori[0], X_be[0], X_sh[0], X_sw[0]]

    for i, ax in enumerate(axs.flatten()):
        ax.imshow(image_samples[i])

    plt.show()
```

In [180…

```
visualize_image_augmentation()
```



4.2 Visualize Model Inference

In [181…

```
def visualize(model_dict):
    """
    Visualize the inference result of the model.
    :param model_dict: Dictionary of the four models.
    """
    # Test Data
    Y_inference = get_labels(data_type="Test")
    X_inference = get_images(data_type="Test", image_names=Y_test[:, 0], flatten=True, resize=True)
    X_inference_noresize = get_images(data_type="Test", image_names=Y_test[:, 0], flatten=False, resize=False)

    # Randomly nine pics
    # Use 3x3 image grid to demonstrate
    num_images = X_inference.shape[0]
    random_indices = random.sample(range(num_images), 9)
    fig, axs = plt.subplots(3, 3, figsize=(15, 15))

    for i, ax in enumerate(axs.flatten()):
        random_index = random_indices[i]

        # Flattened for inference, Original for visualization
        X_sample = np.array([X_inference[random_index]])
        X_sample_noresize = X_inference_noresize[random_index].copy()

        # Prediction & Ground Truth
        predictions = inference(model_dict=model_dict, X=X_sample)
        truths = {
```

```
            "x": Y_inference[random_index, 1],
            "y": Y_inference[random_index, 2],
            "w": Y_inference[random_index, 3],
            "h": Y_inference[random_index, 4],
        }

        ellipse_center_prediction = int(predictions["x"]), int(predictions["y"])
        ellipse_axilen_prediction = int(predictions["w"]), int(predictions["h"])
        cv2.ellipse(
            img=X_sample_noresize,
            center=list(ellipse_center_prediction),
            axes=list(ellipse_axilen_prediction),
            angle=0,
            startAngle=0,
            endAngle=360,
            color=(255, 0, 0),
            thickness=5,
        )

        ellipse_center_truth = int(truths["x"]), int(truths["y"])
        ellipse_axilen_truth = int(truths["w"]), int(truths["h"])
        cv2.ellipse(
            img=X_sample_noresize,
            center=list(ellipse_center_truth),
            axes=list(ellipse_axilen_truth),
            angle=0,
            startAngle=0,
            endAngle=360,
            color=(0, 255, 0),
            thickness=5,
        )

        # Plot
        # Also, show prediction values & legend.
        description = (
            f"Image: {random_index + 1}/{num_images}\n"
            f"Pre: x={predictions["x"]:.1f}, y={predictions["y"]:.1f}, "
            f"w={predictions["w"]:.1f}, h={predictions["h"]:.1f}\n"
            f"Tru: x={truths["x"]:.1f}, y={truths["y"]:.1f}, "
            f"w={truths["w"]:.1f}, h={truths["h"]:.1f}"
        )
        ax.set_title(description, fontsize=10, color="black", loc="center")

        fig.legend(handles=[mlines.Line2D([], [], color='red', linestyle='-', linewidth=2, label='Prediction'),
                            mlines.Line2D([], [], color='green', linestyle='-', linewidth=2, label='Ground Trutl
                   loc='upper center',
                   ncol=2, fontsize=12, frameon=True)

        ax.imshow(X_sample_noresize)
        # ax.set_title(f"Image: {random_index + 1}/{num_images}")

    plt.tight_layout()
    # plt.subplots_adjust(top=0.9)
    plt.show()
```

In [135…

```
with open(os.path.join(model_path, "best_model_rfr_1733156550837645.sav"), "rb") as best_model_rfr_f:
    best_model_rfr_loaded = pickle.load(best_model_rfr_f)

with open(os.path.join(model_path, "best_model_svr_1733156169715915.sav"), "rb") as best_model_svr_f:
    best_model_svr_loaded = pickle.load(best_model_svr_f)
```

In [136…

```
best_model_rfr_loaded
```

Out[136…

```
{'x': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                       n_estimators=30),
 'y': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                       n_estimators=30),
 'w': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                       n_estimators=30),
 'h': RandomForestRegressor(max_depth=19, min_samples_leaf=6, min_samples_split=8,
                       n_estimators=30)}
```

In [137…

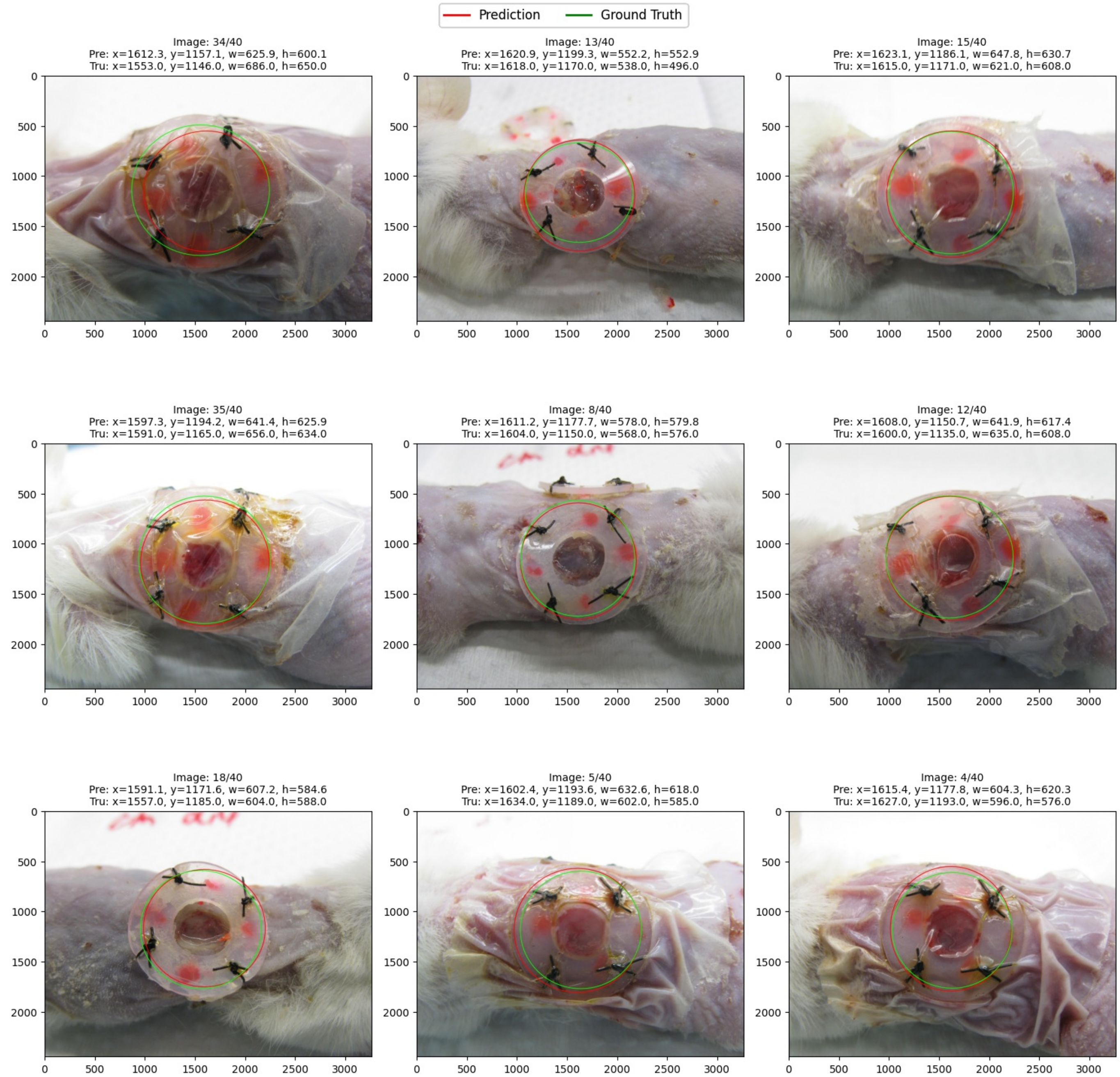
```
best_model_svr_loaded
```

Out[137…

```
{'x': SVR(C=100, epsilon=1),
 'y': SVR(C=100, epsilon=1),
 'w': SVR(C=100, epsilon=1),
 'h': SVR(C=100, epsilon=1)}
```

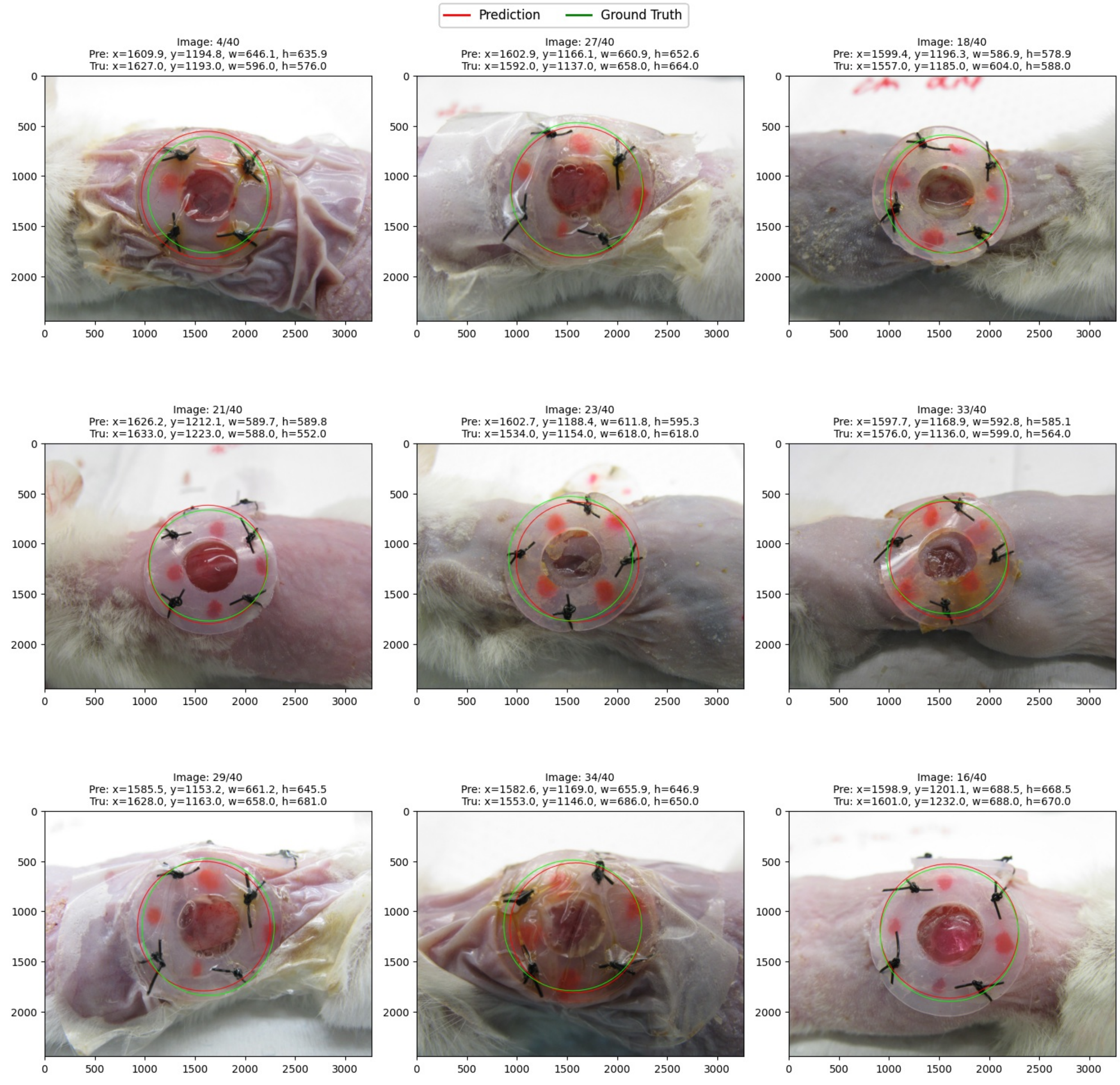
In [182…

```
visualize(best_model_rfr_loaded)
```



In [183…

```
visualize(best_model_svr_loaded)
```


Prediction
Ground Truth
Image: 4/40
Pre: x=1609.9, y=1194.8, w=646.1, h=635.9
Tru: x=1627.0, y=1193.0, w=596.0, h=576.0
Image: 27/40
Pre: x=1602.9, y=1166.1, w=660.9, h=652.6
Tru: x=1592.0, y=1137.0, w=658.0, h=664.0
Image: 18/40
Pre: x=1599.4, y=1196.3, w=586.9, h=578.9
Tru: x=1557.0, y=1185.0, w=604.0, h=588.0
Image: 21/40
Pre: x=1626.2, y=1212.1, w=589.7, h=589.8
Tru: x=1633.0, y=1223.0, w=588.0, h=552.0
Image: 23/40
Pre: x=1602.7, y=1188.4, w=611.8, h=595.3
Tru: x=1534.0, y=1154.0, w=618.0, h=618.0
Image: 33/40
Pre: x=1597.7, y=1168.9, w=592.8, h=585.1
Tru: x=1576.0, y=1136.0, w=599.0, h=564.0
Image: 29/40
Pre: x=1585.5, y=1153.2, w=661.2, h=645.5
Tru: x=1628.0, y=1163.0, w=658.0, h=681.0
Image: 34/40
Pre: x=1582.6, y=1169.0, w=655.9, h=646.9
Tru: x=1553.0, y=1146.0, w=686.0, h=650.0
Image: 16/40
Pre: x=1598.9, y=1201.1, w=688.5, h=668.5
Tru: x=1601.0, y=1232.0, w=688.0, h=670.0