1 Project Description

The main idea of this project is to predict the dependent variable (coordinates of the middle point of the box containing the mouse), based on a set of independent variables (coordinates of 7 joints of the mouse and the coordinates of the 2-point containing box).This is done by designing an automated model based on an artificial neural network.The training dataset is generated from the given images by annotations a set of points for each mouse in the images.

2 Data Analysis

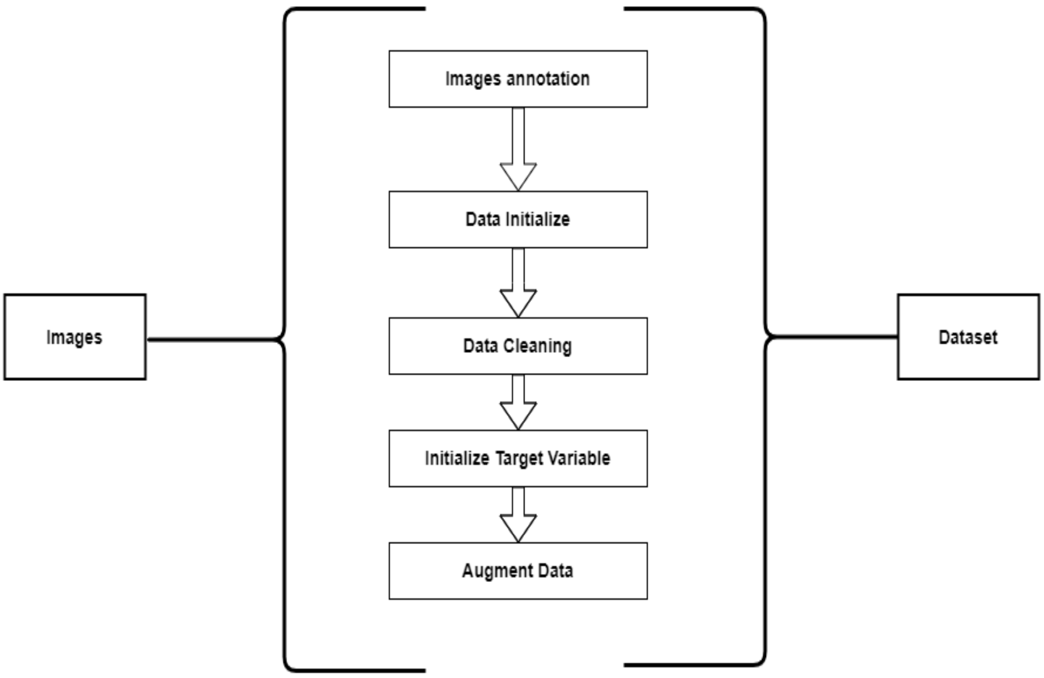

Figure 1: Stage of creating,processing and analyzing data

Figure (1) shows the stage of creating, processing and analyzing data. After that, sequential steps are taken in order to create a suitable data set for building a neural network model.Below we provide a brief summary of each of these steps.

2.1 Images Annotation

The training data set is created from the given images by using the annotation tool to locate 7 joints (nose, left ear, right ear, left hip, right hip, base of the tail, end of the tail), in addition to specifying two angles (top left, lower right) from Bounding box With the mouse inside this bounding box.

The output of this task is two files (CollectedData_annotation.csv , CollectedData_annotation.h5) that contain the same information and coordinates of the specified points.I chose the CollectedData_annotation.h5 file to deal with it in the rest of the data analysis tasks.

2.2 Data Initialize

The CollectedData_annotation.h5 file is loaded as shown in Figure (2).

1. Load notations .

```
df = pd.read_hdf(path)
df.head()
```

		scorer	annotation								
		individuals	mouse1								
		bodyparts	topleft		rightdown		nose		leftear		righ
		coords	x	y	x	y	x	y	x	y	x
labeled-data	0004	A_male_in_a_new_cage_face_view_3_2022-08-10_15-39-01_117.png	163.239390	163.388924	354.817358	355.670287	NaN	NaN	200.900977	175.174665	
		A_male_in_a_new_cage_face_view_3_2022-08-10_15-39-01_137.png	176.858704	181.098197	313.698513	283.183201	NaN	NaN	NaN	NaN	287
		A_male_in_a_new_cage_face_view_3_2022-08-10_15-39-01_266.png	280.215629	64.309673	522.938424	340.171742	410.539804	72.383012	NaN	NaN	
		A_male_in_a_new_cage_face_view_3_2022-08-10_15-39-01_794.png	278.684928	172.456684	420.454752	242.268629	410.177933	211.436922	NaN	NaN	391
		A_male_in_a_new_cage_side_view_4_2022-08-10_15-39-03_1072.png	107.013370	184.790366	236.165308	288.327620	NaN	NaN	NaN	NaN	

5 rows × 36 columns

Figure 2: CollectedData_annotation.h5

We note that the contents of the h5 file are not well formatted, for the purpose of ease of analysis and data handling.I arranged the data and stored it in a data frame, as shown in Figure (3), the characteristics are a set of coordinates (X, Y) for 9 points, and thus the number of characteristics is $9 * 2 = 18$,In addition to the name of the image, the total number of properties will be 19.

2. Initialize dataset function.

This function arranges the data and stores it in a data frame, the purpose of this is for ease of analysis and dealing with the data.

```
def Initialize_dataset_func(df,columns_name):
    data=pd.DataFrame(columns=columns_name)
    mouses=[df[(df.columns.levels[0][0],df.columns.levels[1][i])]  for i in range(len(df.columns.levels[1]))]
    for mouse in mouses:
        for index,img in enumerate(list(mouse.index.droplevel([0,1]))):
            row={'image_name':img}
            for point in mouse.iloc[index].index:
                name=point[0]+'_'+point[1]
                row[name]=mouse.iloc[index][point[0]][point[1]]

            data=data.append(row, ignore_index=True)

    return data

#.........................................
```

```
columns_name=['image_name','topleft_x','topleft_y','rightdown_x','rightdown_y',
             'nose_x','nose_y','leftear_x','leftear_y','rightear_x','rightear_y'
             ,'leftHip_x','leftHip_y','rightHip_x','rightHip_y','tailBase_x',
              'tailBase_y','tailEnd_x','tailEnd_y']

#.........................................

data=Initialize_dataset_func(df,columns_name)
data
```

	image_name	topleft_x	topleft_y	rightdown_x	rightdown_y	nose_x	nose_y	leftear_x	leftear_y	rig
0	A_male_in_a_new_cage_face_view_3_2022-08-10_15...	163.239390	163.388924	354.817358	355.670287	NaN	NaN	200.900977	175.174665	
1	A_male_in_a_new_cage_face_view_3_2022-08-10_15...	176.858704	181.098197	313.698513	283.183201	NaN	NaN	NaN	NaN	28
2	A_male_in_a_new_cage_face_view_3_2022-08-10_15...	280.215629	64.309673	522.938424	340.171742	410.539804	72.383012	NaN	NaN	
3	A_male_in_a_new_cage_face_view_3_2022-08-10_15...	278.684928	172.456684	420.454752	242.268629	410.177933	211.436922	NaN	NaN	39
4	A_male_in_a_new_cage_side_view_4_2022-08-10_15...	107.013370	184.790366	236.165308	288.327620	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
245	A_male_meet_with_the_same_cage_mate_top_view_1...	424.969557	126.791945	535.991675	305.832644	432.737525	294.598772	475.655214	288.782823	46
246	A_male_meet_with_the_same_cage_mate_top_view_1...	99.935963	113.083088	254.528892	268.514354	107.412621	125.781814	115.760980	149.542527	14
247	A_male_meet_with_the_same_cage_mate_top_view_1...	418.994990	66.994948	616.036324	259.826441	600.860502	79.535826	555.182042	88.492387	57
248	A_male_meet_with_the_same_cage_mate_top_view_1...	404.409393	125.198777	617.928178	278.654516	605.338782	238.962606	565.034259	215.675549	56
249	A_male_meet_with_the_same_cage_mate_top_view_1...	494.196541	261.168700	611.087857	358.641788	524.729736	344.650022	585.634348	306.136811	54

250 rows × 19 columns

Figure 3: Initialize dataset

We note that each record of the data represents a mouse, and it can be more than one record with the same name as the image.

2.3 Data Cleaning

I cleaned the data by deleting all records that contain at least one empty point.Since it is not logical to use methods for processing null values in the data, for example, we cannot use average points.Therefore, in order to build an accurate model, it is better to delete these records, as shown in the Figure (4).

3. Clean data function.

This function cleans the data by deleting all records that contain at least one empty point. Since it is not logical to use methods for processing null values in the data, for example, we cannot use average points. Therefore, in order to build an accurate model, it is better to delete these records.

In [31]:
```
def Clean_data_func(data):
    data_clean=data.loc[~np.isnan(data['topleft_x'])]
    data_clean=data_clean.dropna(axis=0)

    return data_clean

#.........................................
```

In [32]:
```
data_cleand=Clean_data_func(data)
data_cleand
```

Out[32]:

	image_name	topleft_x	topleft_y	rightdown_x	rightdown_y	nose_x	nose_y	leftear_x	leftear_y	rig
15	A_male_in_a_new_cage_top_view_1_2022-08-10_15-...	128.549357	65.820937	343.968291	200.680474	132.377104	77.059977	154.363355	95.197897	16
18	A_male_in_a_new_cage_top_view_1_2022-08-10_15-...	182.812086	143.602228	283.598977	347.408787	259.303180	335.523716	252.674845	305.475251	22
20	A_male_in_a_new_cage_top_view_1_2022-08-10_15-...	323.281931	157.519263	470.461324	335.410553	381.936249	324.591375	418.334613	298.639752	38
21	A_male_in_a_new_cage_top_view_1_2022-08-10_15-...	26.225608	191.685630	183.768734	313.040632	74.518446	308.047809	70.616530	269.063588	3
22	A_male_in_a_new_cage_top_view_1_2022-08-10_15-...	329.321661	185.938033	546.875231	299.702867	535.775526	290.910586	532.401728	254.460285	51
...	...	...	...	...	...	...	...	...	...	
243	A_male_meet_with_the_same_cage_mate_top_view_1...	173.344676	235.319293	409.774519	341.718369	405.137659	314.445055	367.843597	294.286103	36
244	A_male_meet_with_the_same_cage_mate_top_view_1...	299.291235	226.168538	459.190583	340.394053	452.083636	324.270671	430.531201	297.744598	42
245	A_male_meet_with_the_same_cage_mate_top_view_1...	424.969557	126.791945	535.991675	305.832644	432.737525	294.598772	475.655214	288.782823	46
247	A_male_meet_with_the_same_cage_mate_top_view_1...	418.994990	66.994948	616.036324	259.826441	600.860502	79.535826	555.182042	88.492387	57
248	A_male_meet_with_the_same_cage_mate_top_view_1...	404.409393	125.198777	617.928178	278.654516	605.338782	238.962606	565.034259	215.675549	56

67 rows × 19 columns

Figure 4: Data cleaned

When deleting empty records, the data decreased from 250 to 67 records.After that, a set of pictures is shown, and dots and boxes are drawn on the mice, as shown in the Figure (5).

```
np.random.seed(42)
j=1
plt.figure(figsize=(10,7))
for i in np.random.randint(0,len(images_test),4):
    plt.subplot(2,2,j)
    Show_image_func(images_path,images_test[i],data_cleand,Pcolor='red',Rcolor='green')
    j=j+1
plt.show()
```

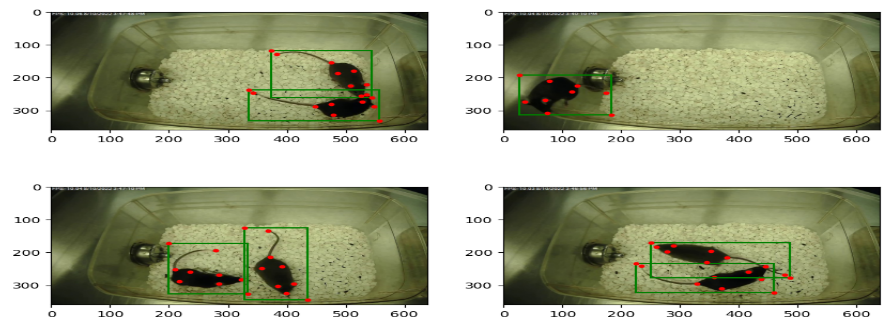

Figure 5: samples

2.4 Initialize Target Variable(Dependent Variable)

The dependent variable, which will be predicted by the machine learning model to be designed, represents the midpoint of the mouse enclosure.Since the dependent variable is not present in the data set, the dependent variable (the midpoint of the box) was formed through the two points $(topleft, rightdown)$ .as in equation (1).

$$
\begin{aligned}
center_x &= \frac{topleft_x + rightdown_x}{2} \\
center_y &= \frac{topleft_y + rightdown_y}{2}
\end{aligned}
\tag{1}
$$

By applying equation (1), the dependent variable $(topleft, rightdown)$ is created in the data set file as shown in Figure (6).

```
In [46]: def Initialize_target_variable(data):
             datanew=copy.deepcopy(data)
             center_x=(data.topleft_x.values+data.rightdown_x.values)/2
             center_y=(data.topleft_y.values+data.rightdown_y.values)/2
             datanew['center_x']=center_x
             datanew['center_y']=center_y
             return datanew

         #.........................................
```

```
In [47]: data_end=Initialize_target_variable(data_cleand)
         data_end
```

nose_y	leftear_x	leftear_y	rightear_x	...	leftHip_x	leftHip_y	rightHip_x	rightHip_y	tailBase_x	tailBase_y	tailEnd_x	tailEnd_y	center_x	center_y
77.059977	154.363355	95.197897	167.807133	...	201.120920	150.993897	213.729179	129.843165	229.917556	149.815392	332.780504	187.137346	236.258824	133.250705
335.523716	252.674845	305.475251	229.917556	...	225.595863	253.800711	198.693663	267.843947	206.515111	233.040787	248.008505	148.887415	233.205531	245.505508
324.591375	418.334613	298.639752	389.658766	...	451.601816	253.949019	417.024036	244.412569	430.181703	222.638718	346.434878	164.380057	396.871627	246.464908
308.047809	70.616530	269.063588	36.025450	...	116.980847	242.020260	77.787093	210.681555	125.234024	223.809760	173.479478	247.216590	104.997171	252.363131
290.910586	532.401728	254.460285	519.045633	...	468.441760	222.545986	455.720861	256.838787	432.002286	225.369593	337.339689	192.599096	438.098446	242.820450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
314.445055	367.843597	294.286103	369.187527	...	293.460309	311.509892	296.384232	340.326952	260.665165	324.188549	181.709268	251.280337	291.559597	288.518831
324.270671	430.531201	297.744598	422.241803	...	366.332095	274.570108	349.569969	300.099847	331.058427	266.659356	349.709572	218.995319	379.240909	283.281295
294.598772	475.655214	288.782823	463.822767	...	528.720730	258.339341	501.593007	247.309094	519.776202	223.202990	462.619467	132.153314	480.480616	216.312295
79.535826	555.182042	88.492387	574.053347	...	527.270525	161.354010	546.433340	171.851775	526.521048	178.953650	434.268473	243.440887	517.515657	163.410695

Figure 6: Data set after initialize target variable

The dependent variable is displayed on a set of images, as shown in Figure (7). The blue dots are the dependent variables that will be predicted by the machine learning model that will be designed based on a set of independent variables represented in the red dots.

```
pcolors=['red']*((data_end.shape[1]-1)//2-1)
pcolors.append('blue')
np.random.seed(42)
j=1
plt.figure(figsize=(10,7))
for i in np.random.randint(0,len(images_test),4):
    plt.subplot(2,2,j)
    Show_image_func(images_path,images_test[i],data_end,Pcolor=pcolors,Rcolor='green')
    j+=1
plt.show()
```

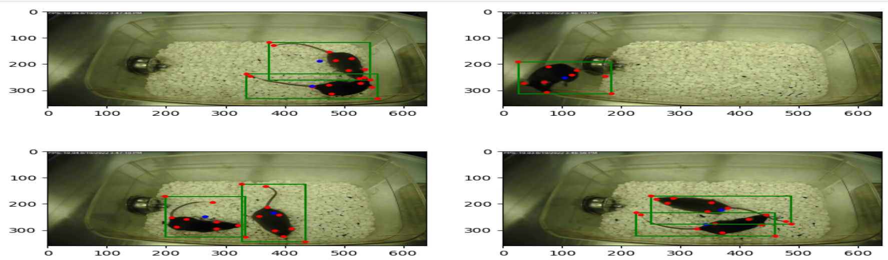

Figure 7: Dependent variables in the Images

2.5 Augment Data

After deleting the NaN rows, our total number of records decreased from 250 to 67.So we need to create new images to increase the number of data.Since our work depends on a set of points only, therefore we do not need to produce new images, but we need to produce new points i.e. mouse New), and in order to increase the number of rats I simply applied geometric transformations to the rat points in the data set, by applying the rotation transform by angles $(30, 60, 90, 120, 150, 180, 210, 240, 270, 300)$ to the points of each mouse in the data set, I used homogeneous matrices to perform the rotation transformation As follows.

$$P^{(org)} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_{10} & y_{10} & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix} \tag{2}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -x_c\cos\theta + y_c\sin\theta + x_c & -x_c\sin\theta - y_c\cos\theta + y_c & 0 \end{bmatrix}$$

$$P^{(new)} = P^{(org)} * T \tag{3}$$

The previous rotation matrix is applied to each mouse in the data set at different angles.This means that from each mouse in the original dataset, 10 mice with different orientations will be produced.

In the end, we obtained 737 mouse (record) as our final data set which we will use to build a neural network model.

3 Structure of machine learning model

Neural network models support multi-output regression and have the advantage of continuous function learning that can model a more graceful relationship between changes in input and output. Multi-output regression can be supported directly by neural networks simply by specifying how many target variables are present in the problem as the number of nodes in the output layer.This task contains two output variables (x, y) output layer of a neural network with two nodes in the output layer, each of which has a linear activation function.

We will define a multilayer model (MLP) for the multi-output regression task defined in this project , Since the data set is small, it is a good practice to use k-fold cross validation to evaluate several MLP models in our multiple-output regression tasks. The number of nodes and layers in the model can easily be adapted and customized according to the complexity of the data set. I built two multi-layer (MLP) models for the multi-output regression task which differ in the number of hidden layers and the number of nodes in each layer, as Figure (8) shows the structure of the models.

5.1 built Models.

```
In [10]: def Create_Model_One(n_inputs, n_outputs):
             model = Sequential()
             model.add(Dense(18, input_dim=n_inputs, kernel_initializer='normal', activation='relu'))
             model.add(Dense(9, activation='relu'))
             model.add(Dense(n_outputs))
             model.compile(loss='mse', optimizer='adam',metrics=['mse'])
             return model

         #..................................................................

         def Create_Model_Tow(n_inputs, n_outputs):
             model = Sequential()
             model.add(Dense(18, input_dim=n_inputs, kernel_initializer='normal', activation='relu'))
             model.add(Dense(9, activation='relu'))
             model.add(Dense(4, activation='relu'))
             model.add(Dense(n_outputs))
             model.compile(loss='mse', optimizer='adam',metrics=['mse'])
             return model

         #..................................................................
```

Figure 8: Structure models

- Each sample has 18 inputs and 2 outputs, thus, networks require an input layer that expects 18 inputs defined via the 'input_dim' argument in the first hidden layer and two nodes in the output layer.
- I used the ReLU activation function that is common in hidden layers. The first model contains two hidden layers, the first layer contains 18 nodes and the second layer contains 9 nodes, while the second model contains three hidden layers, the first containing 18 nodes, the second containing 9 and the third containing 4. The number of layers and the number of nodes in these models were chosen after some trial and error.
- We will fit the models using the mean square error (MSE) because it depends on the loss of $L2$, and Adams version of the random gradient origin.
- I evaluated these models using k-fold cross validation with 10 times and three iterations in order to choose the best model.

  Models are defined, fit and evaluated on each fold, MSE scores are collected and can be summarized by reporting the mean and standard deviation , The following results were noted:

```
In [14]: sc={'Model_One MSE':results_model_one,
             'Model_Tow MSE':results_model_tow
            }
         pd.DataFrame(data=sc).describe().T

         #..................................................................
```

Out[14]:

	count	mean	std	min	25%	50%	75%	max
Model_One MSE	30.0	0.000151	0.000048	0.000091	0.000119	0.000136	0.000162	0.000307
Model_Tow MSE	30.0	0.006511	0.011967	0.000084	0.000142	0.000178	0.009104	0.037817

Through the previous evaluation, it is clear that the first model is the best, because the mean is 0.000151, and the standard deviation is 0.000048 for the MSE, which is much less than the other model.

`Once we have chosen an appropriate model configuration, we can use it to fit a final model on all available data and make predictions on new data.`

```
In [15]: model=Create_Model_One(X_train.shape[1],y_train.shape[1])
```

Figure 9: Evaluated-models using k-fold cross validation

Through the previous evaluation, it is clear that the first model is the best, because the mean is 0.000151, and the standard deviation is 0.000048 for the MSE, which is much less than the other model.

Once we have chosen an appropriate model configuration, we can use it to fit a final model on all available data and make predictions on new data.

4 Experiment results

In the experiment of predicting the midpoint of the mouse box, the data set is increased by performing rotation transformation with different angles.Finally, 737 samples are obtained, which are then divided into the training set 80% and the test set with 30%, after that the best model is identified and evaluated .In terms of the number of layers and the number of nodes, by using k-fold cross validation with 10 times and three iterations, it was noted that the best model is the first model that was designed, which achieved an average MSE of 0.000151 with a standard deviation of 0.000048.

After we choose the appropriate configuration of the model, it is trained on the available training data set and evaluated on the test data set, the model is trained with loss mse, epoch 300, optimizer adam .The changes of the mse cost function were observed during the training process, as shown in Figure (10).

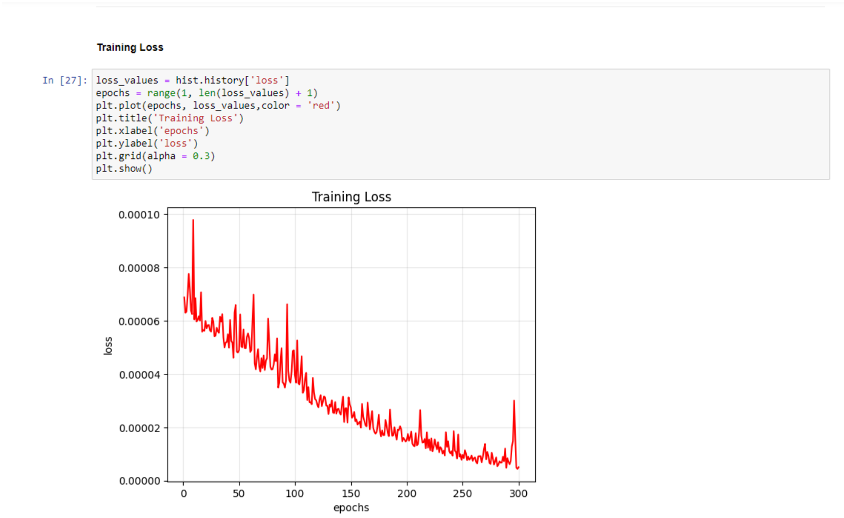

Figure 10: Training loss

We conclude from the previous figure that the training phase was somewhat better, because mse decreases in each epoch.

The model was evaluated using the test data set, the model achieves a mean square error (MSE) of $9.742738257045858e-06$.

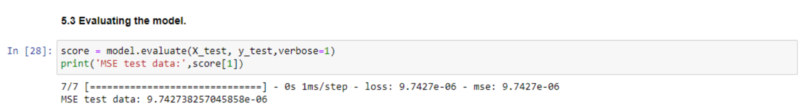

Figure 11: Evaluate model

As Figure (12) shows a comparison between the points of the predicted centers and the real centers in the test data set, we notice that the prediction was better.

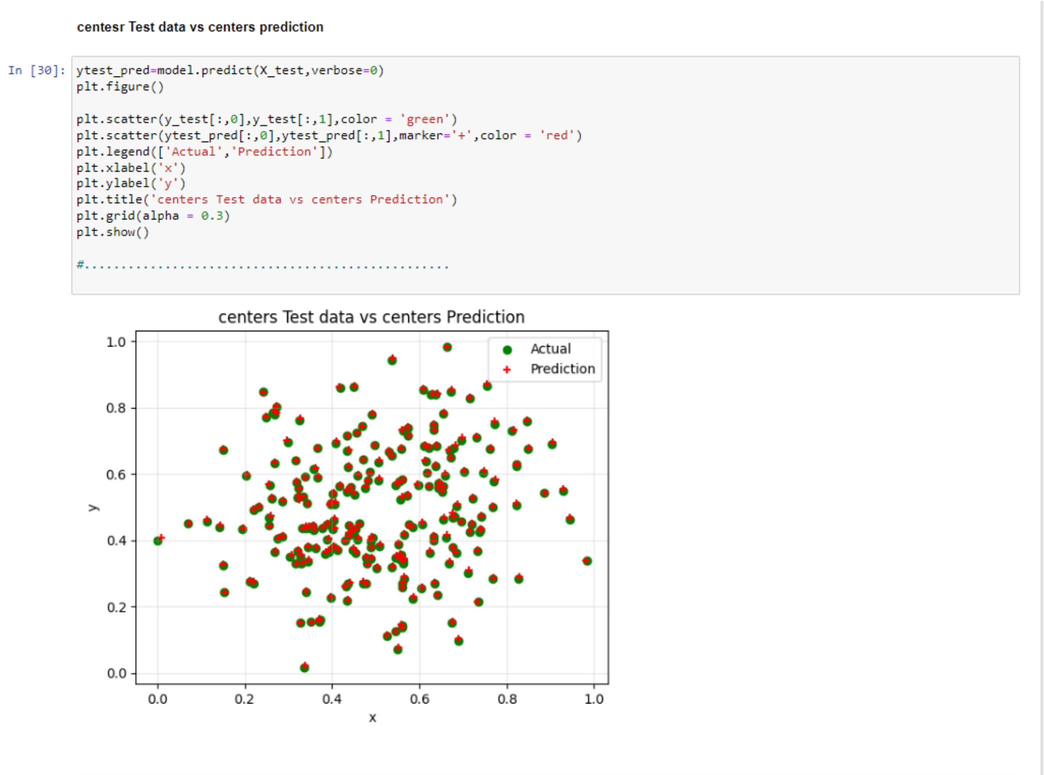

Figure 12: Centers Test data vs Centers prediction

We also evaluated for each of x,y coordinates, as shown in Figure (13).

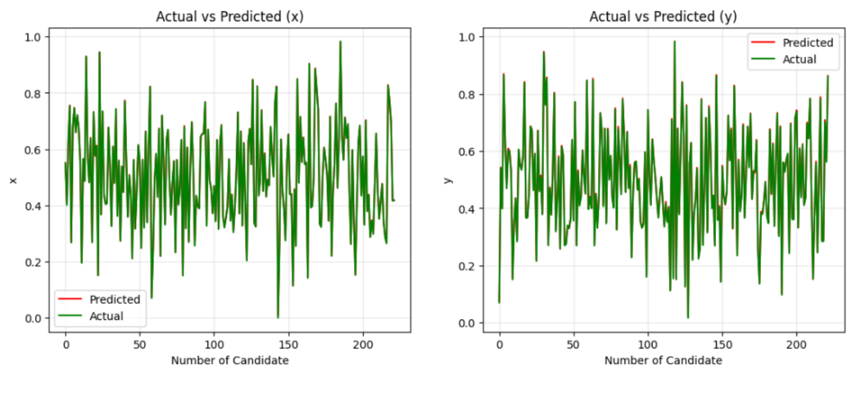

Figure 13: Evaluated (x,y) coordinates

We conclude from the previous figure that the model predicts both the x and y events well.

5 Instructions for running code

The project code consists of two files "ipynb"

- Initialize_dataset_section.ipynb

  This file is for the stage of data handling and analysis.

- built_model_section.ipynb This is a file for the process of building and analyzing the performance of the designed models.

- The "Initialize_dataset_section.ipynb" file is first run in order to perform all operations related to the original data set, and the new data set resulting from the implementation of this code is saved in the "data" folder named "initialize_dataset.csv".

- Then the "built_model_section.ipynb" file is executed, which uses the dataset generated from the previous file to build neural network models.And the trained model is saved as "my_model.h5"

Note: The codes in the two files were built using functions, and these functions are called when needed.This is for the purpose of organization and ease of dealing with it.It also contains a lot of annotations. Each file can be run using "Restart & Run All" and the results can be observed after completion of the execution

6 Conclusions

As we all know, deep learning relies on a large amount of data to train the model.Therefore, rotational transformation was used to address the lack of training data, although the data was increased from 67 to 737, but it is not sufficient to generalize a model based on neural networks. The data set was normalized using MinMaxScaler, after that it was divided into the 80% training set and the 30% test set.

Since the data set is small and in order to avoid overfitting problems of the MLP models (number of layers, number of nodes), k-fold cross validation with 10 times and three iterations was used in order to evaluate two multilayer models (MLP) for the multiple output regression task. The activation function was used.ReLU common in hidden layers.The first model contains two hidden layers, the first layer contains 18 nodes and the second layer contains 9 nodes, while the second model contains three hidden layers, the first contains 18 nodes, the second contains 9 and the third contains 4.Use the mean squared error (MSE) for the fit of these models because it is based on L2 loss and the Adams version of Stochastic Gradient Origin.

The first model is the best, because the mean is 0.000151, and the standard deviation is 0.000048 for the MSE which is much lower than the other model.

And after our proper selection of the model, it is trained on the available training data set and evaluated on the test data set, the model is trained with mse loss, epoch 300, adam optimizer.We conclude that the training phase was rather good. The model was evaluated using the test data set, where the model achieved a mean square error (MSE) of $9.742738257045858e - 06$.