**CrowdWise: Predictive Modeling for Gym Crowdedness using Regression**

**Submitted for**

**INTELLIGENT MODEL DESIGN USING AI**

Submitted by:

**(E23MCAG0006) VIJAY BHARGAV BHAMIDI**

**(E23MCAG0083) YASH KANANI**

Submitted to

**DR. NITIN ARVIND SHELKE**

**Jan-May 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sr.No | Content | Page No |
|  |  |  |
|  |  |  |

1. **ABSTRACT**

In the current generation, youth spend most of their time in the gym trying keep up their fitness. But the problem arises when a person goes to the gym but all the gym machinery that they require are already occupied and they have to wait till the person using it finishes thus wasting a lot of time. This paper brings a solution to this problem by using many intelligent models that predict the correct time for a person to visit the gym based on the parameters in the dataset used. In this paper we have used possible regression techniques and advanced regression techniques using ensemble learning.

1. **Introduction**

Within the busy campus of a university, the gymnasium is a center for exercise and well-being. It is essential to comprehend and forecast the movement of people into the university gymnasium at any given moment in order to guarantee effective resource distribution. Our goal in this research is to create an advanced prediction model that is especially suited to the dynamics of a university gymnasium.

Equipped with an extensive dataset that spans more than a year of gymnasium usage, we explore the nuances of predicting gym attendance, taking into account a variety of variables such as the number of attendees, the day and time of the event, the weather, academic calendar events like the beginning and middle of the semester, and other relevant aspects specific to university life. Our goal is very clear: to develop a predictive model that can accurately predict gym occupancy levels while also identifying the underlying factors influencing these trends.

To accomplish this goal, we use a complex approach based on advanced regression algorithms and ensemble learning procedures. Combining the strength of ensemble learning strategies like bagging and boosting with algorithms like random forests, decision trees, support vector machine (SVM) and linear regression. Our goal is to build a strong predictive framework that can manage the complexity included in the data from university gymnasiums. Furthermore, by employing meticulous feature engineering and selection procedures, we hope to extract the most significant predictors from the wide range of data available to us, highlighting the key variables influencing gym attendance.

1. **Related Work**

Crowd counting is and always has been a very challenging task. It always posted problems like large gathering of crowd and also low gathering crowd along with several other obstructions. Jia Wan et.al. proposed the method of using a residual regression framework that uses the correlation information between the samples [3]. All the existing crowd counting algorithms rely heavily on the histogram-based features to calculate the crowd properties. Then regression is used to calculate the crowd size [1]. In the paper by Antoni B. Chan, et.al. size of inhomogeneous crowds having pedestrians that are travelling in different directions were calculated using Bayesian Regression [4]. Crowd counting is used to estimate the number of people present in the crowd it is a challenge as there are various degree congestions and people’s appearances are taken into consideration. In the paper by Tan Xin et.al. this problem was addressed using the approach of Multi-layer Regression Network (MRNet) [2]. Unsupervised learning is the less taken road to count people. The paper by Yuting Liu, et. al. explored an approach where they detect and count people using an unlabeled target set by transferring bi-knowledge learnt from regression- and detection-based models in a labeled source set [5]. R. Wang, et. al. proposed a self-supervised learning framework with unlabeled and limited labeled data for pre-training and fine-tuning crowd counting model (SSL-FT) [6].

|  |  |
| --- | --- |
| *References* | * David Ryan, et.al (2015) * Tan, Xin, et al. (2019) * Wan, J., et.al. (2019) * A. B. Chan, et. al. (2012) * Yuting Liu, et. al. (2020) * R. Wang, et.al. (2023) |
| *Methods/Technologies/Languages used* | * Programming Language used: Python Programming Language * Libraries Used:   -> Pandas: Pandas is a powerful and open-source Python library. The Pandas library is used for data manipulation and analysis. Pandas consist of data structures and functions to perform efficient operations on data.   -> Scikit-Learn: Scikit-Learn is an open-source Python library that implements a range of machine learning, pre-processing, cross-validation, and visualization algorithms using a unified interface. |
| *Dataset used* | Crowdedness at a Campus Gym:  https://www.kaggle.com/datasets/nsrose7224/crowdedness-at-the-campus-gym |
| *Performance Metrics* | Mean Absolute Error:  Mean Square Error:  R2 Score:  RMSE: |
| *Details/Limitations* | This method is unable to reverify the predictions made as correct or not. This would have allowed the model to report a continuously improving accuracy as the data would have been updated at every prediction. |

**Table1:** Literature Overview

1. **Methodology**

In this paper we are proposing an experimental approach by which we are intending to apply max voting ensemble learning on all the regression models. The following procedures are followed to achieve our final output.

* 1. **Data Collection:**

The dataset consists of 26,000 people counts (about every 10 minutes) over the last year [1 (dataset)]. In addition, gathered extra info including weather and semester-specific information that might affect how crowded it is. The Label (independent variable) is the number of people, which we like to predict given some subset of the Features (dependent variable).

Label:

• Number of people

Features:

• *date* (string; datetime of data)

• *timestamp* (int; number of seconds since beginning of day)

• *day\_of\_week* (int; 0 [Monday] - 6 [Sunday])

• *is\_weekend* (int; 0 or 1) [Boolean, if 1, it's either Saturday or Sunday]

• *is\_holiday* (int; 0 or 1) [Boolean, if 1 it's a federal holiday, 0 otherwise]

• *temperature* (float; degrees Fahrenheit)

• *is\_start\_of\_semester* (int; 0 or 1) [Boolean, if 1 it's the beginning of a school semester, 0 otherwise]

• *month* (int; 1 [Jan] - 12 [Dec])

• *hour* (int; 0 - 23)

* 1. **Data Preprocessing:**

We imported the NumPy, pandas, and pytz libraries for preprocessing the dataset. Pandas library is used for loading and preprocessing of the dataset. The dataset is initialized with the variable *data*. The dataset exploration part involved examining the first five rows of the dataset using *data.head()* and checking the dimensions of the dataset with *data.shape()*.

Duplicate entries in the dataset were identified by using *data.duplicated().sum()* and missing values were addressed by using the *data.isnull().sum()*. The columns that were unnecessary were dropped using *data.drop()*.Extraction of independent variable and dependent variables were done using *data.iloc()* and stored as *Y(independent variable)* and *X(dependent variables)*.

Date and time were extracted from the datetime column using *pd.to\_datetime()* and are divided into year, day, minutes and seconds. The dataset was then split into training and testing subsets, with 75% allocated for training and 25% for testing. The training data of dependent variables were stored in variable *X\_train* and the training data of independent variables were stored in *Y\_train*.

* 1. **Model Selection:**

In this proposed methodology, we have applied regression models to analyze and anticipate the number of individuals present at the gymnasium at a specific hour of the day, taking all the information of *X* into consideration. We have applied the following regression models.

1. Linear Regression: It is one of the easiest and most early regression models which finds linear/polynomial relationship between *X* and *Y*. We have used the following parameters with their multiple values and loop over them to find best hyper parameter which suits the dataset.

To arrive at the best parameter and its value we used:

*bestparavalue = {}*

*for parameter, values in params:*

*bestMetrics=♾️*

*for value in values:*

*Linear Regression()*

*training*

*testing*

*performance metrics*

*if mae < bestmae:*

*update best\_value[mae]=value*

*if mse < bestmse:*

*update best\_value[mse]=value*

*if rmse < bestrmse:*

*update best\_value[rmse]=value*

*if r2 > bestR2:*

*update best\_value[r2]=value*

*if prediction\_time < besttime:*

*update best\_value[prediction\_time]=value*

*bestparavalue[parameter] = best\_value*

After we got the best parameter and their respective values we re-run the Linear Regression model with just those parameters combined which gives us the best possible performance/accuracy from the model and dataset.

Then the final model is used to get prediction of multiple input data.

1. Support Vector Regression (SVR): Since SVR does not performs when the dataset has linear relationship which is true in our case so we have nor invested anymore computational resources than required by using just some basic hyper parameters
2. Random Forest Regression (RFR): It is one of the most advanced regression model. which finds linear/polynomial relationship between *X* and *Y*. We have used the following parameters with their multiple values and loop over them to find best hyper parameter which suits the dataset.

Fixed parameters:

To arrive at the best parameter and its value we used:

*bestparavalue = {}*

*for parameter, values in params:*

*bestMetrics=♾️*

*for value in values:*

*Linear Regression()*

*training*

*testing*

*performance metrics*

*if mae < bestmae:*

*update best\_value[mae]=value*

*if mse < bestmse:*

*update best\_value[mse]=value*

*if rmse < bestrmse:*

*update best\_value[rmse]=value*

*if r2 > bestR2:*

*update best\_value[r2]=value*

*if prediction\_time < besttime:*

*update best\_value[prediction\_time]=value*

*bestparavalue[parameter] = best\_value*

After we got the best parameter and their respective values, we re-run the Random Forest Regressor model with just those parameters combined which gives us the best possible performance/accuracy from the model and dataset.

Then the final model is used to get prediction of multiple input data.

1. XGBoost Regressor (XGB): eXtreme Gradient Boosting regressor is one of the powerful boosting ensemble learning technique. Ensemble learning gives the combined prediction of several individual models to give a more accurate prediction. XGBoost uses various boosters as its core. We have used the following three boosters and basic hyper parameters.

To arrive at the best parameter and its value we used:

*for booster in boosters:*

*XGB Regressor()*

*training*

*testing*

*performance metrics*

From the three boosters available the gbtree performs the best considering it also took the least amount of time between these three.

We retrain the model using gbtree and which gives us the best possible performance/accuracy from the model and dataset.

Then the model then is used to get prediction of multiple input data.

1. ADABoost Regressor (ADA): Adaptive Boosting regressor is one of the best ensemble learning technique. It is designed to improve the performance of weak learners by coming them into strong learners.

To arrive at the best parameter and its value we used:

*best\_value\_ADA = {}*

*for estimator in baseEstimators:*

*ADABoost Regressor()*

*training*

*testing*

*performance metrics*

*if mae < bestmae:*

*update best\_value[mae]=value*

*if mse < bestmse:*

*update best\_value[mse]=value*

*if rmse < bestrmse:*

*update best\_value[rmse]=value*

*if r2 > bestR2:*

*update best\_value[r2]=value*

*if prediction\_time < besttime:*

*update best\_value[prediction\_time]=value*

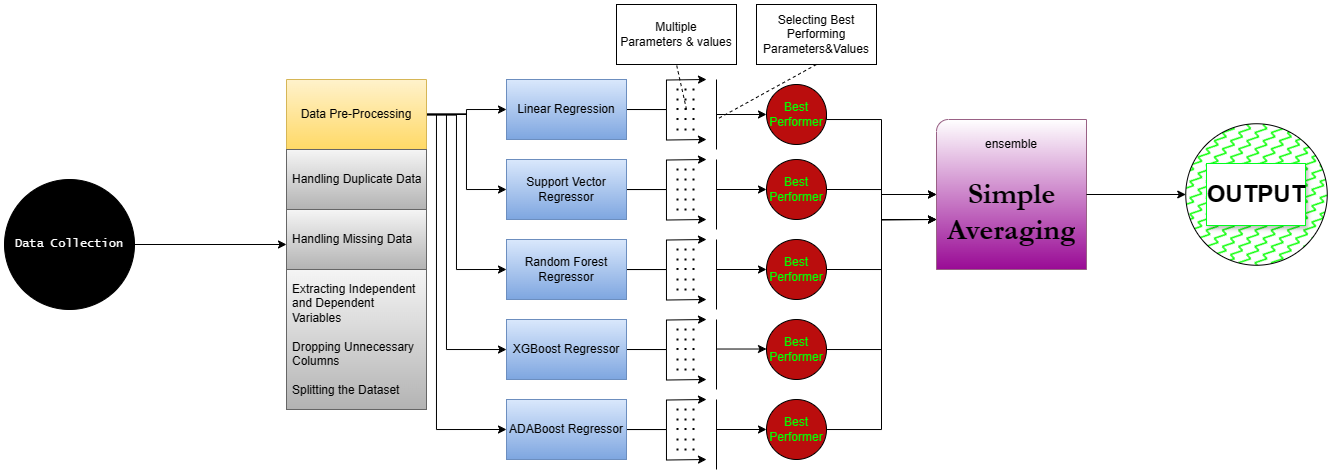
*bestparavalue[parameter] = best\_value*

Then we use the best performer estimator (for R2 in this case) to re train model and use it to form final decission

*adaBest = AdaBoostRegressor(estimator=best\_value\_ADA['R2'])*

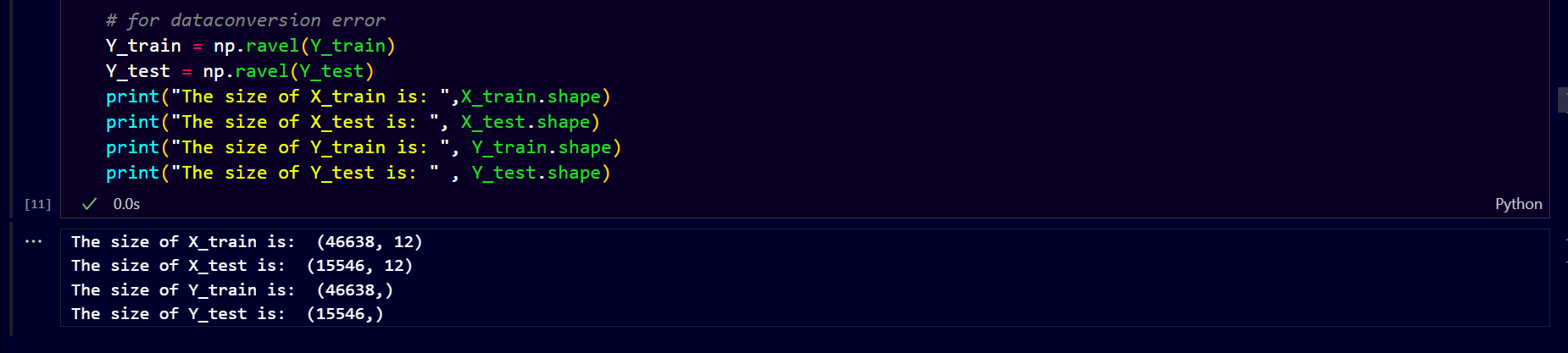
1. Ensemble(simple averaging)

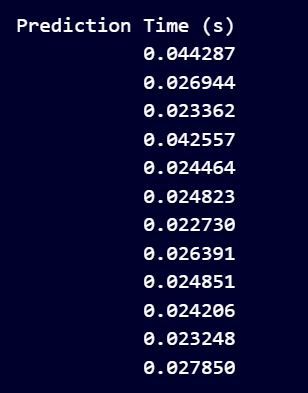
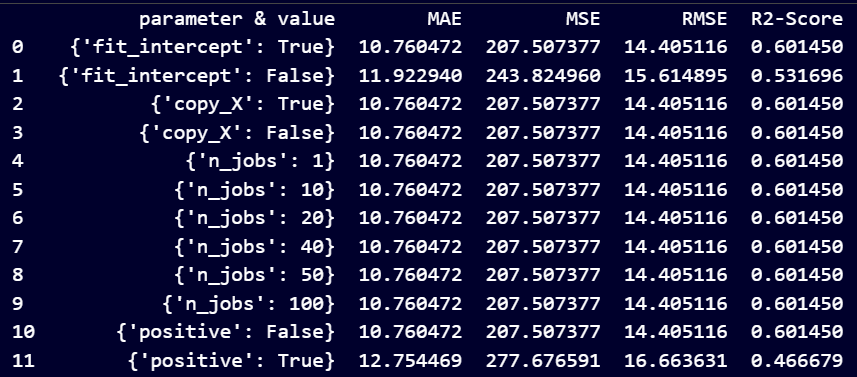
After all the best performers are trained and used to get output we average it using simple averaging technique of ensemble learning to get the final output.

*****(predictionsLR+predictionsRFR+predictionsXGB+predictionsADA)/4*

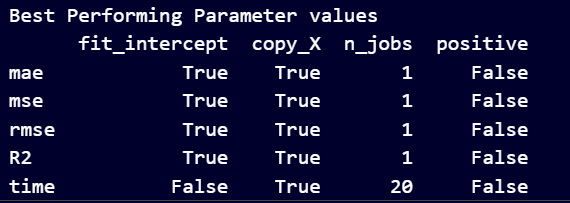
1. **Result and Discussion**

After all the pre-processing and splitting the dataset the format of our training and testing data were

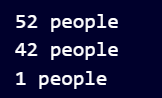


After applying all the linear regression parameters   
  


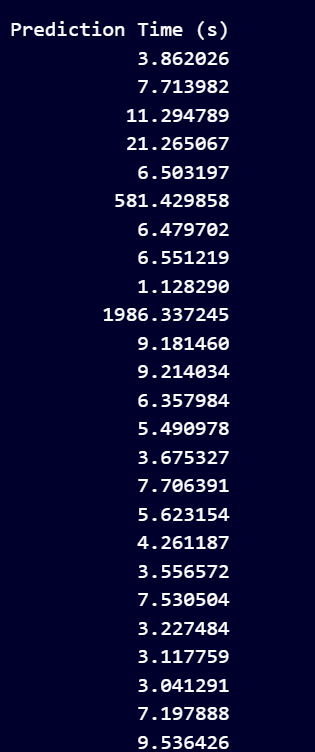
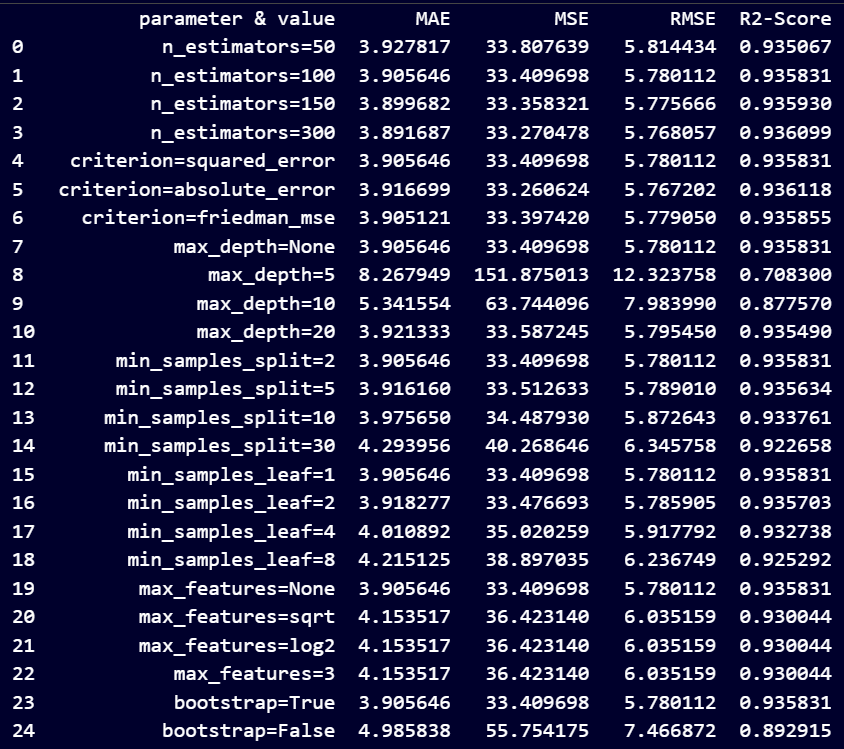
The best performing parameters were



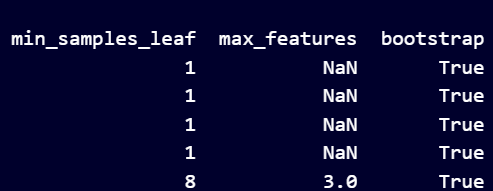
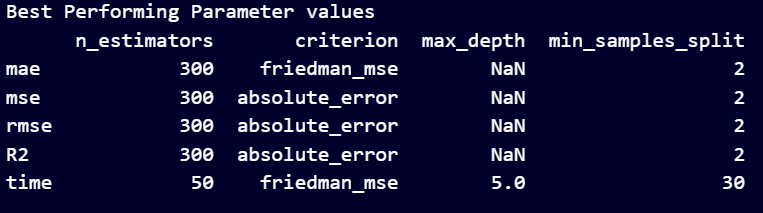
And the results were



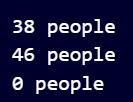
After applying RFR with all the parameters and values



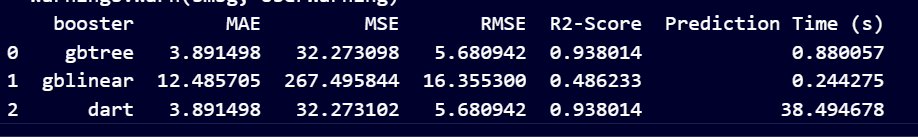
The best performers were



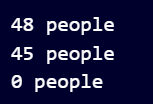
And the results were



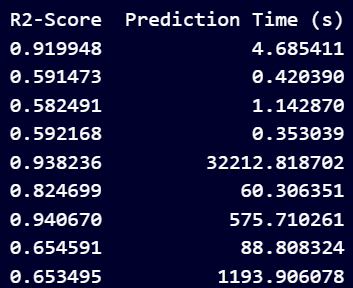
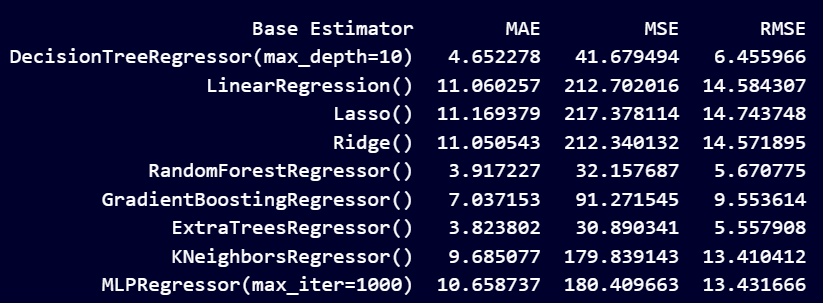
After applying XGBoostReggresor with all the boosters and some fixed parameters



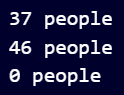
And after selecting best booster the results were



After applying ADA boost regressor with all the possible base estimator and some fixed parameters



And the results after selecting best performer were



1. **Conclusions**

In essence, we used all the regression models as our dataset’s dependent variable was continues.

In all the models we used we applied many parameters and hyperparameters with their value.

We found the best performing parameter and value pair, used it to re train the model whose accuracy was already greater than all the model’s individual parameters.

Then on the best possible output from each model we used ensemble learning’s simple averaging to get even more refined results.

1. **Future Scope**

Instead of using parameters and their values and re running the model again with them, we can improve the proposed method by storing the trained model itself and then using the best performer parameters with its model directly. This will reduce total time taken to give output a lot.

The model is not capable adding data as it goes on *i.e..* the model is not learning new features on its own and we have to update the dataset for it to happen.

**References**

1. David Ryan, Simon Denman, Sridha Sridharan, Clinton Fookes, an evaluation of crowd counting methods, features and regression models, Computer Vision and Image Understanding, Volume 130,2015, Pages 1-17, ISSN 1077-3142, <https://doi.org/10.1016/j.cviu.2014.07.008>.
2. Tan, Xin, et al. "Crowd counting via multi-layer regression." Proceedings of the 27th ACM International Conference on Multimedia. 2019.
3. Wan, J., Luo, W., Wu, B., Chan, A.B. and Liu, W., 2019. Residual regression with semantic prior for crowd counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4036-4045).
4. A. B. Chan and N. Vasconcelos, "Counting People With Low-Level Features and Bayesian Regression," in IEEE Transactions on Image Processing, vol. 21, no. 4, pp. 2160-2177, April 2012, doi: 10.1109/TIP.2011.2172800.
5. Yuting Liu, Zheng Wang, Miaojing Shi, Shin'ichi Satoh, Qijun Zhao, and Hongyu Yang. 2020. Towards Unsupervised Crowd Counting via Regression-Detection Bi-knowledge Transfer. In Proceedings of the 28th ACM International Conference on Multimedia (MM '20). Association for Computing Machinery, New York, NY, USA, 129–137. <https://doi.org/10.1145/3394171.3413825>.
6. R. Wang, Y. Hao, L. Hu, J. Chen, M. Chen and D. Wu, "Self-Supervised Learning With Data-Efficient Supervised Fine-Tuning for Crowd Counting," in IEEE Transactions on Multimedia, vol. 25, pp. 1538-1546, 2023, doi: 10.1109/TMM.2023.3251106.

**GitHub Link**

[YashKanani11/ApplyingNewMethodologyForReggresion (github.com)](https://github.com/YashKanani11/ApplyingNewMethodologyForReggresion)