# K Nearest Neighbors

## Problem Description:

The task aimed at predicting house prices using K-Nearest Neighbors (KNN) regression on a dataset comprising 50,000 records. The goal was to assess the model's predictive capability for housing prices based on various features.

## Data and Preprocessing:

- The dataset was loaded from 'housing\_price\_dataset.csv'

- Features such as 'SquareFeet', 'Bedrooms', 'Bathrooms', 'Neighborhood\_Category', and 'YearBuilt' were selected, while 'Price' served as the target variable.

- The dataset was split into training and testing sets (75% training, 25% testing) using `train\_test\_split`.

- Normalization was performed on the dataset using MinMaxScaler.

## Model Building and Evaluation:

- A KNN regression model (`KNeighborsRegressor`) with `n\_neighbors=5` and `metric='euclidean'` was trained on the training data.

- Predictions were made on both the training and testing sets.

- Evaluation metrics - Root Mean Squared Error (RMSE) - was used to assess model performance:

- Training RMSE: 44663.38

- Testing RMSE: 54512.57

## Scatterplot Visualization and Conclusion:

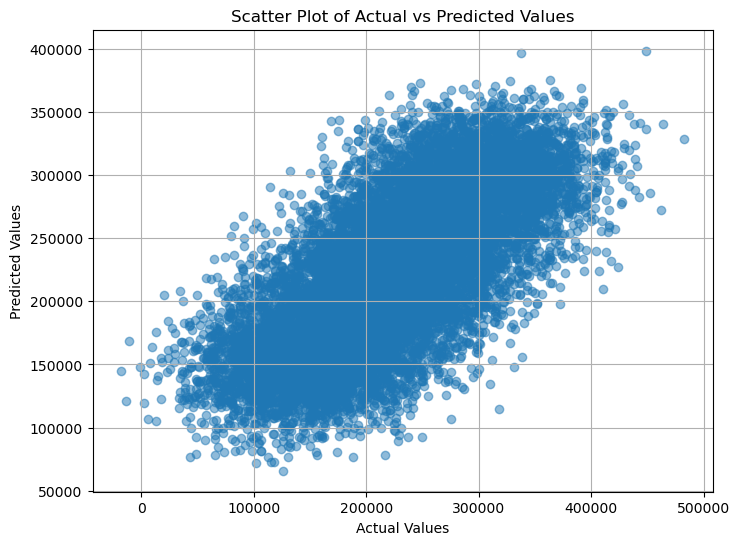
- Visual inspection of the scatterplot showcased a positive correlation between actual and predicted values.

- A concentrated cluster around the diagonal indicated consistent and accurate predictions, with few outliers.

- Training RMSE of $44663.38 and Testing RMSE of $54512.57 indicated model deviations from actual prices, with slightly higher error on the testing set.

- The small difference between Training and Testing RMSE suggested minimal overfitting, indicating reasonable model generalization.

- The KNN regression model showed promising performance in predicting house prices.



# Neural Networks

## Problem Description:

The objective was to predict housing prices using a self-implemented neural network regression model. The dataset consisted of 2,000 samples, including features such as 'SquareFeet', 'Bedrooms', 'Bathrooms', 'Neighborhood\_Category', and 'YearBuilt', with 'Price' being the target variable.

## Data Preprocessing:

- The dataset was loaded and down sampled to 2,000 samples.

- Features and target variables were isolated, and the dataset was split into training, validation, and test sets (70% training, , 15% validation, 15% testing) using `train\_test\_split`.

- The shapes of each set were displayed to confirm the split.

## Neural Network Implementation:

- A neural network architecture with input size(5), hidden layer(1), neuron size(5), and output size(1) was established.

- Normalization was applied to the target variable for training, testing and validation sets.

## Training the Neural Network:

- Two weight initialization methods, 'zeros' and 'random', were employed.

- Training was conducted using the `train` function, evaluating losses and applying early stopping based on validation loss.

- Learning rate comparisons were visualized for both zero and random weight initialization methods.

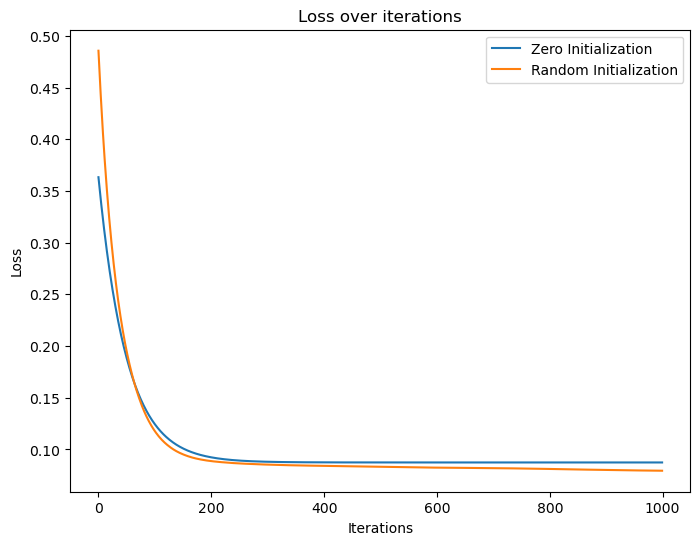


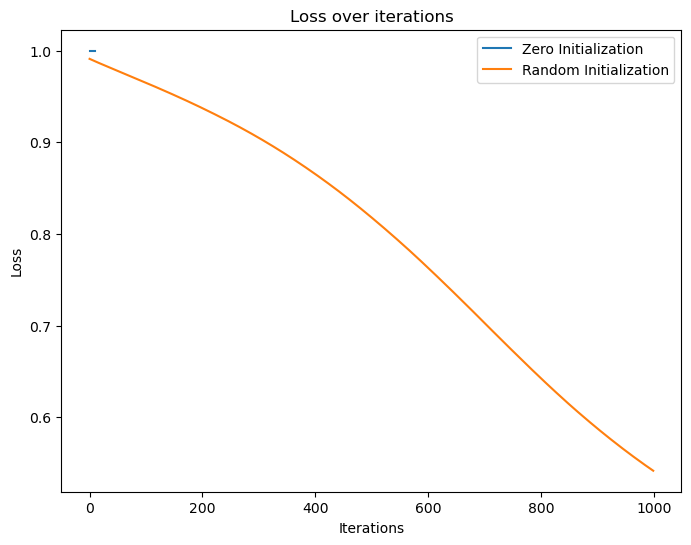
Figure Random Weights Performing slightly better without Early Stopping

Figure Early Stopping at Epoch 10 for Zero Weights

## A graph of different colored lines Description automatically generated

Figure LR 0.001 performs slightly better in Random Initialization

## Model Evaluation:

- The neural network models were evaluated using Mean Squared Error (MSE) and R-squared (R2) on both training and testing datasets.

- Results showed differing performances between zero and random initialization methods.

- Zero Initialization:

- Training MSE: 0.99, R2: 0.0

- Testing MSE: 0.97, R2: -0.01

- Random Initialization:

- Training MSE: 0.54, R2: 0.45

- Testing MSE: 0.50, R2: 0.47

## Conclusion:

- The evaluation highlighted that both models underperform compared to the expected accuracy.

- The zero initialization model exhibited poor performance on both training and testing datasets.

- Conversely, the random initialization model showed relatively better performance, capturing some patterns and performing better on unseen data compared to the zero initialization model.

- Further adjustments in the neural network architecture, hyperparameters, or employing different training strategies might be necessary to enhance predictive performance.