Word Count: 2948

CI702 APPLIED AI REPORT

What is the functionality of your system?

The system is designed to predict the prices of property in the Boston area using the Boston housing dataset which is based on input features. These features include socio-economic, environmental, and housing-related data such as the crime rate in an area, average number of bedrooms, proximity to the city centre, and demographic.

By creating a model that uses a regression-based approach to analyse the dataset, the model tries to identify patterns and relationships between the input features and house prices. Depending on the models chosen, it allows it to provide accurate price predictions for unseen data, offering worthwhile insights and allowing real estate agents or a potential buyer to confidently estimate the real value of the property.

The system integrates functionalities that allow it to perform effectively. It preprocesses the raw data by overseeing missing values, scaling features, and encoding categorical variables where necessary. During the training phase, it builds a regression model using different methods specified such as Linear Regression, Random Forest, or an ensemble model. It also validates its performance with metrics such as root mean squared error and R^2 values allowing me to judge how successful it has been in its predictions.

What are the basic inputs and outputs?

The system takes features as inputs which describe various characteristics of residential areas in the city of Boston. These features can include numerical and categorical data and there are a total of 13 features which include the following:

- CRIM: Crime rate per capita.
- ZN: The Proportion of residential land zoned for plots over 25,000 sq. feet.
- INDUS: Proportion of non-retail business acres.
- CHAS: A binary variable indicating proximity to the Charles River (=1 if located close to the river and 0 if otherwise).
- RM: Average number of rooms per house.
- AGE: Proportion of owner-occupied units built prior to the 1940s.
- DIS: Weighted distance to five Boston employment centres.
- RAD: Index of accessibility to orbital motorways.
- TAX: Full value property tax rate per $10,000.
- PTRATIO: Pupil-teacher ratio.
- AAPOP: A measure related to the proportion of African American residents (This is an ethical issue and is discussed later in the report).
- LSTAT: The percentage of lower level educated residents.

The main output of this system is the median value (MEDV) of a property for a given set of input features. This output provides an estimate of the property's median value enabling a potential buyer to give the best price based on all the features the model has processed.

What is the real-world problem that this solves?

The system is designed to address the real-world issue of accurately estimating housing prices. The Boston housing dataset is an outdated dataset, which can be seen from its features. However, it is still a good place to start and to assess the model to see if it can accurately predict property prices based on specific features that people look for when selecting a house to buy. This problem is a major one that younger generations globally are facing, from millennials to gen z; the value of property is quickly rising to unreachable values. If we take London as an example, between Jan 2014 and Jan 2024 the price of all properties in the Barking and Dagenham area saw an increase in price of 78.2% from £184,884 to £329,413 on average [1].

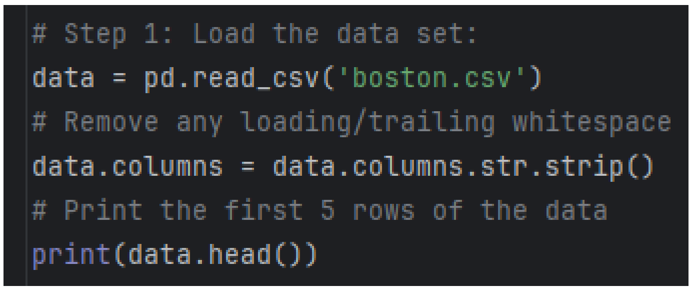
A model can only do so much, but it is still a good guide into what affects the market value of properties. For first time buyers, the model can serve as a tool to help understand the trends in the market and help them make better financial decisions. For councils and urban planners, it can help them identify which areas of the city are of more concern and allow them to allocate resources to help improve those areas. The reasons for that could be poor infrastructure, high crime rates, or lower level of higher educated individuals in that area. The system should not have any bias, but it can be introduced based on the features in the dataset. In my opinion, having a column such as the number of African American residents per 1000 citizens is an ethical issue as it reinforces the negative stereotypes of the perception that people have of these areas in cities.

To summaries this section, the model eliminates extra work for people and companies looking to valuate properties by offering an efficient and understandable solution to accurately predict housing prices. The next step for this would be to use a more up to date dataset and one that is more relevant to me such as a Brighton or London house price dataset.

Details regarding algorithm selection

General Prerequisites for all Algorithms

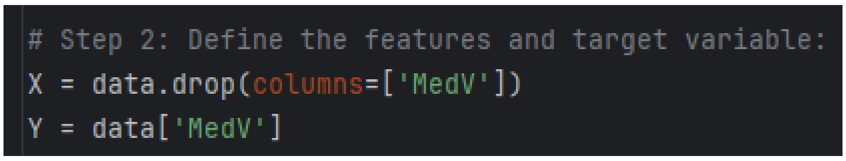
For all algorithms it was necessary to first load the dataset and trim any whitespaces that might be present in the dataset.



```
# Step 1: Load the data set:
data = pd.read_csv('boston.csv')
# Remove any loading/trailing whitespace
data.columns = data.columns.str.strip()
# Print the first 5 rows of the data
print(data.head())
```

FIGURE 1: LOADING DATASET AND STRIPPING WHITESPACES

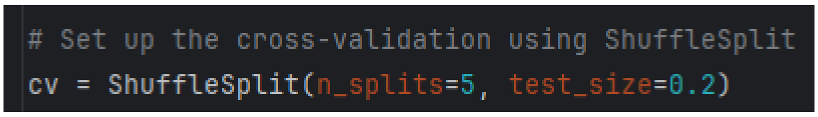
For the next step I had to define the features and the target variable. To do this the MedV column was dropped from the dataset and if we had categorical data that was not numeric by nature then it would have been necessary to encode those features. Thankfully, there was no such data present in this dataset.



```
# Step 2: Define the features and target variable:
X = data.drop(columns=['MedV'])
Y = data['MedV']
```

FIGURE 2: SELECTING THE CORRECT OUTPUT FEATURE

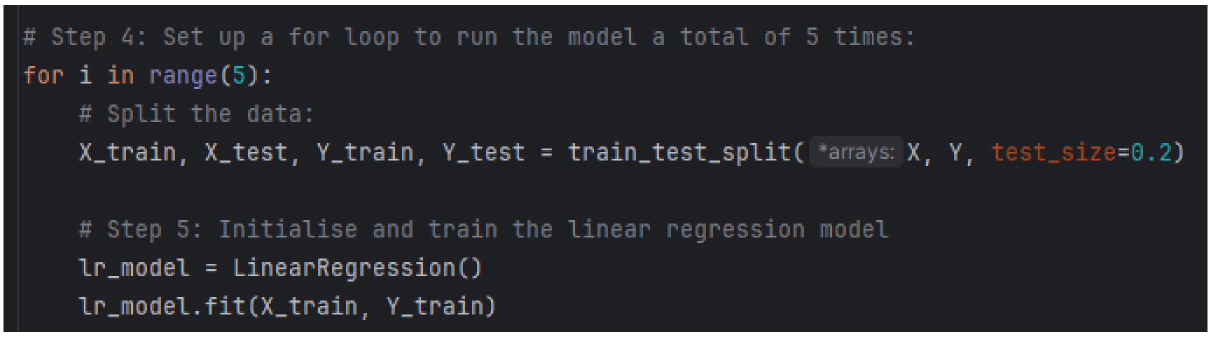
It was also necessary to setup cross-validation to analyse the effectiveness of the model. This was done using shuffle split. Shuffle split is a cross-validation strategy that shuffles the data like a pack of cards every time the model is run [2]. For this dataset, the data has been split into 5 different train and test pairs, each with a random selection of samples. It then uses 80% of the data to train itself and then the remaining 20% for testing purposes to see if it can accurately predict property values.



```
# Set up the cross-validation using ShuffleSplit
cv = ShuffleSplit(n_splits=5, test_size=0.2)
```

FIGURE 3: CROSS-VALIDATION USING SHUFFLE SPLIT

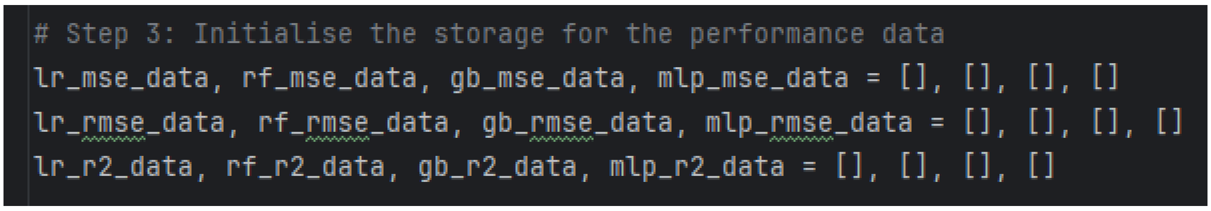
To make this model easier to gather data for, I decided to use a for loop to gather data for 5 runs at once. To do this, I made sure that the training and prediction making sections of the code were all inside the for loop.



```
# Step 4: Set up a for loop to run the model a total of 5 times:
for i in range(5):
    # Split the data:
    X_train, X_test, Y_train, Y_test = train_test_split( *arrays: X, Y, test_size=0.2)

    # Step 5: Initialise and train the linear regression model
    lr_model = LinearRegression()
    lr_model.fit(X_train, Y_train)
```

FIGURE 4: INITIALISING THE ALGORITHMS INSIDE THE FOR LOOP

But for the data to be saved from these runs, I had to create empty arrays that stored all models Mean Squared Error, Root Mean Squared Error and R^2 values.

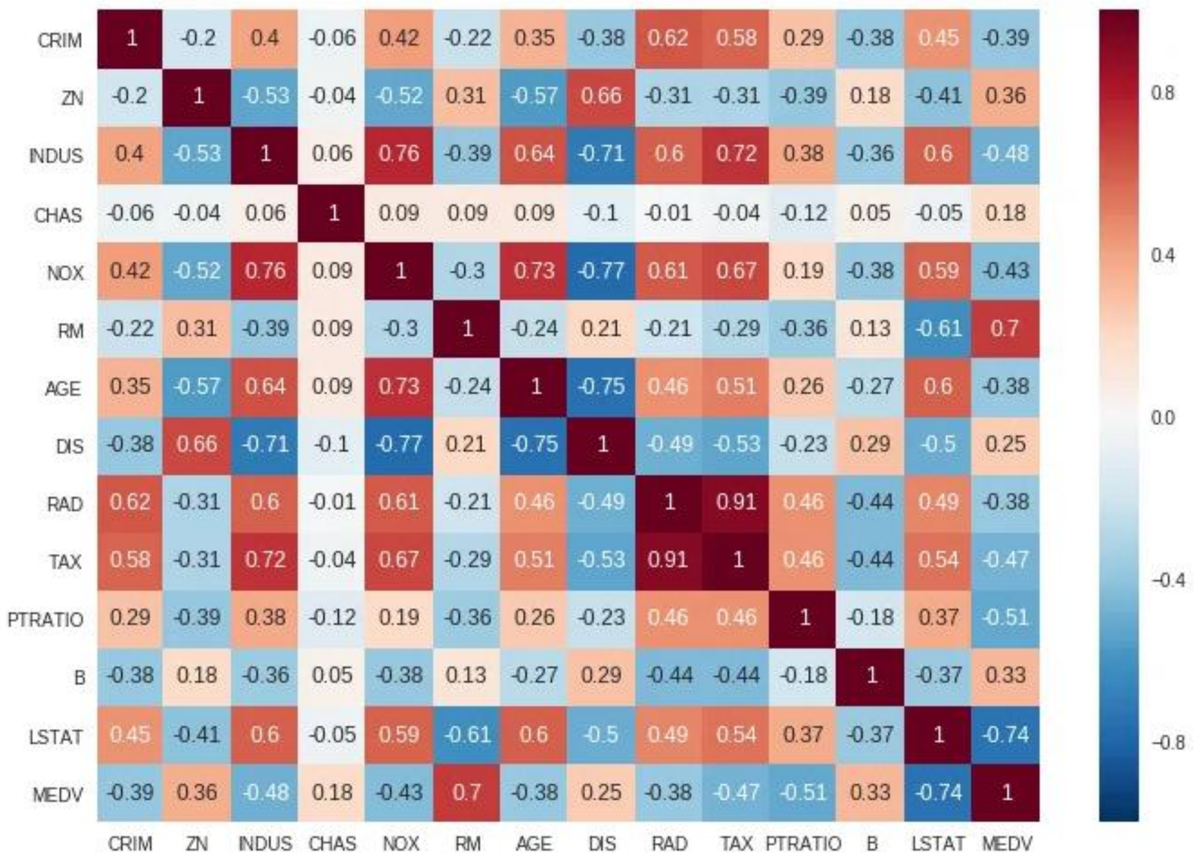


```
# Step 3: Initialise the storage for the performance data
lr_mse_data, rf_mse_data, gb_mse_data, mlp_mse_data = [], [], [], []
lr_rmse_data, rf_rmse_data, gb_rmse_data, mlp_rmse_data = [], [], [], []
lr_r2_data, rf_r2_data, gb_r2_data, mlp_r2_data = [], [], [], []
```

FIGURE 5: ARRAYS TO STORE DATA IN FOR EVERY RUN

Linear Regression

The first algorithm that I tried was the Linear Regression algorithm. The Boston housing dataset is not perfectly linear, but it does show traits of being linear. Firstly, there is a positive correlation between the median value of properties and the number of rooms it has. However, there is also a negative correlation when comparing the LSTAT to the median value [3].



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	-0.2	0.4	-0.06	0.42	-0.22	0.35	-0.38	0.62	0.58	0.29	-0.38	0.45	-0.39
ZN	-0.2	1	-0.53	-0.04	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
INDUS	0.4	-0.53	1	0.06	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
CHAS	-0.06	-0.04	0.06	1	0.09	0.09	0.09	-0.1	-0.01	-0.04	-0.12	0.05	-0.05	0.18
NOX	0.42	-0.52	0.76	0.09	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
RM	-0.22	0.31	-0.39	0.09	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
AGE	0.35	-0.57	0.64	0.09	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
DIS	-0.38	0.66	-0.71	-0.1	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
RAD	0.62	-0.31	0.6	-0.01	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
TAX	0.58	-0.31	0.72	-0.04	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
PTRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
B	-0.38	0.18	-0.36	0.05	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
LSTAT	0.45	-0.41	0.6	-0.05	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
MEDV	-0.39	0.36	-0.48	0.18	-0.43	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

FIGURE 6: CORRELATION COEFFICIENT RANGES FOR FEATURE VARIABLES [3]

Linear regression algorithms are simple and easy to interpret and have a wide application. They also work well with continuous data such as the output of this dataset, the median value. Linear regression is also less prone to overfitting and given that the dataset is small with few features at only 506 entries and 13 features, overfitting is less likely to occur [4]. With linear regression models it is better to have a larger dataset to avoid overfitting, but this could also affect how it categorises the data because of too many details and noisy data [5]. I used the linear regression algorithm as a base to compare other algorithms with discussed below.

Random Forest Regressor

The random forest regressor is a supervised learning algorithm that combines multiple decision trees made from randomly chosen subsets of the data to make an overall prediction for unseen data points [6]. It is a good algorithm to process larger datasets and capture more complex connections and correlations than a singular decision tree. The random forest algorithm is useful for predicting continuous data such as the median value of properties and is particularly good at capturing non-linear relationships between input and target features, such as the Boston housing dataset [6].

Random forest regressor tends to overfit less compared to other regressor algorithms because it constructs multiple trees independently and then averages their results [6]. This approach makes random forest regression more dependable than traditional linear models, especially for more complex problems [6].

Gradient Boosting Regressor

I used the gradient boosting regressor because it is a similar ensemble algorithm to the random forest algorithm and thus wanted to compare the performance of the two. The gradient boosting regressor works slightly different to the random forest as instead of building independent trees, it builds them sequentially [7]. Each new tree corrects the mistakes of the previous tree, so every new iteration of that tree works to minimise the residual errors [7].

This sounds better than the random forest regressor right? However, since the trees are built sequentially, the algorithm is prone to overfitting especially with noisy data and a high number of iterations [7]. The use of computer resources and time can be heavier as each tree depends on the previous one so it will take longer to train this algorithm. It is also more challenging to optimise and requires careful tuning with hyperparameters [7].

Gradient boosting is especially well suited to applications in the healthcare industry as it is better at predicting the likelihood of a disease to occur in an individual such as heart disease or diabetes [8]. This is because it has a better ability at handling complex data and still achieve a higher level of accuracy [7]. Healthcare data often involves non-linear relationships between distinctive features and the gradient boosting algorithm can notice these subtle patterns better than other methods such as the random forest [8].

Multi-Layer Perceptron (MLP) Regressor

Multi-Layer perceptron's are a network of neurons that can be used in regression problems and consist of an input layer, one or more hidden layers and then an output layer [9]. MLPs are particularly good at identifying non-linear interactions between variables and features so I believe that using the MLP regressor for the Boston housing dataset would work well. The dataset has many non-linear interactions between features such as the number of rooms, crime rates or property age and the MLP will be better suited at capturing these relationships better than the Linear Regression algorithm which assumes a linear relationship.

There is also potential for further hyperparameter tuning with this algorithm and I am hoping that if it is optimised correctly, it will have the potential to outperform the traditional algorithms such as the Random Forest or Gradient Boosting Regressors.

How have I measured if my system is successful?

To measure success in this project I was focusing on achieving an R^2 value higher than 80% which would mean that 80% of the predictions are correct. It is a statistical way to measure how good the regression line approximates the actual data [10]. I also used mean absolute error as this is a good measure for a model in machine learning because the absolute error refers to the magnitude of difference between the prediction of an observation and the true value of that observation [11]. The lower the number the better the closer to the actual value the prediction is. Finally, I used the root mean squared error to assess my model as the RMSE is a metric that tells us the average difference between the predicted values and the actual values in a dataset [12]. So, the smaller the number the more accurate the prediction is.

Cross-Fold Validation is also a useful metric to observe as the above results give us an average but observing the results for each fold can tell us if a model is stable. Too much variation across the fold means that the model is not performing at its best. For my model we have split the data into 5 folds meaning each iteration will exclude 20% of the dataset to test with and the other 80% will be used to train with [13].

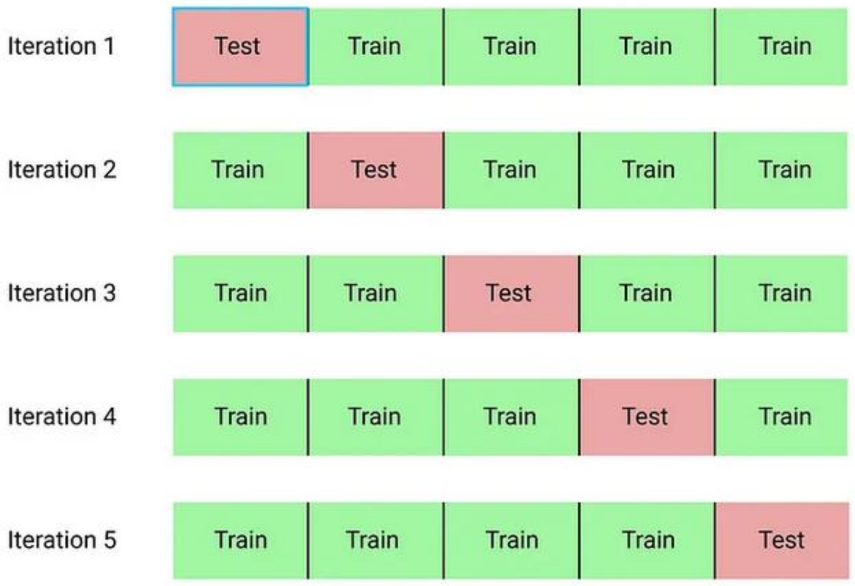


FIGURE 7: 5-FOLD CROSS-VALIDATION [13]

Discussion of results

Baseline models with no tuning

The median value is based on $1000 so, a value of 25 is equal to $25,000. This is the same for all the other metrics [14]. For the first set of results, I have left all the algorithms as they are with no tuning whatsoever and we get the following results in Table 1:

TABLE 1: SINGLE RUN RESULTS FOR MODELS WITHOUT ANY TUNING

	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
MAE	3.6995	2.3160	2.1427	4.3335
RMSE	5.1382	3.4846	2.7940	5.7933
R^2	0.6157	0.8233	0.8864	0.5115

It can already be seen that the linear regression algorithm falls behind in all results but specifically the R^2 and mean squared error values compared to the random forest and gradient boosting algorithms. This indicates that the Boston housing dataset is not a linear dataset and there are more complex relationships in the dataset that need to be explored to get a better prediction. The mean absolute error is higher than the random forest and gradient boosting models at 3.7 which is equal to $3,700 and the R^2 value is 0.6157 meaning the model can explain 61.6% of the variance in the dataset. The MLP regressor fails on all metrics when compared to the other models but, this is expected as it has not been tuned yet.

In Table 2 below we see the average results for cross-validation. For linear regression, all results are better than the single split result indicating that on average the model performs reasonably well and serves as a good benchmark for the other models to build upon. The random forest model also sees improvement across the board and outperforms the linear regression model and, the gradient boost sees a very slight decline in results but nothing significant. However, the MLP model sees a significant decline in results on average indicating that the single run may have been luck.

TABLE 2: AVERAGE CROSS-VALIDATION RESULTS FOR MODELS WITHOUT TUNING

	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
CV-MAE	3.4277	2.2816	2.1679	5.0864
CV-RMSE	4.7130	2.8626	3.0547	6.5446
CV-R^2	0.7290	0.8314	0.8705	0.4413

The graphs below show the results for each different run and where the average values for the folds were obtained from.

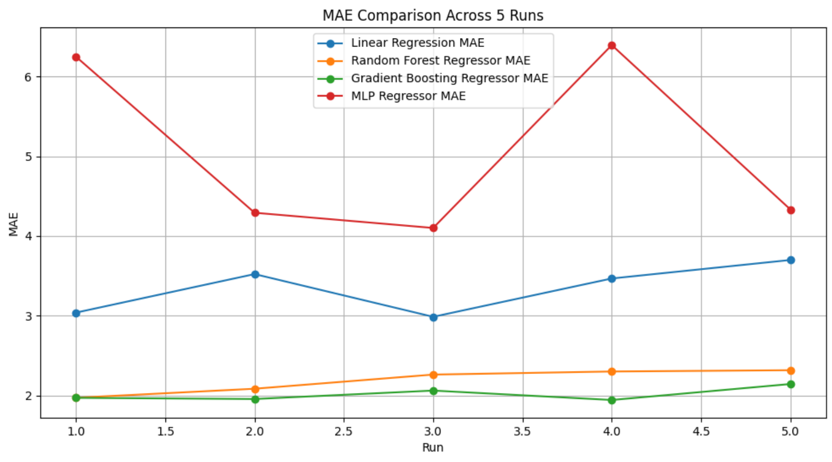


FIGURE 8: MAE COMPARISON ACROSS 5 RUNS

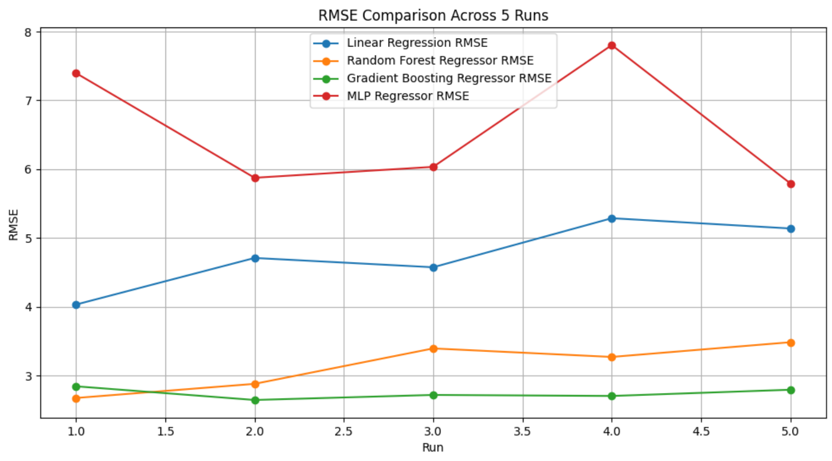


FIGURE 9: RMSE COMPARISON ACROSS 5 RUNS

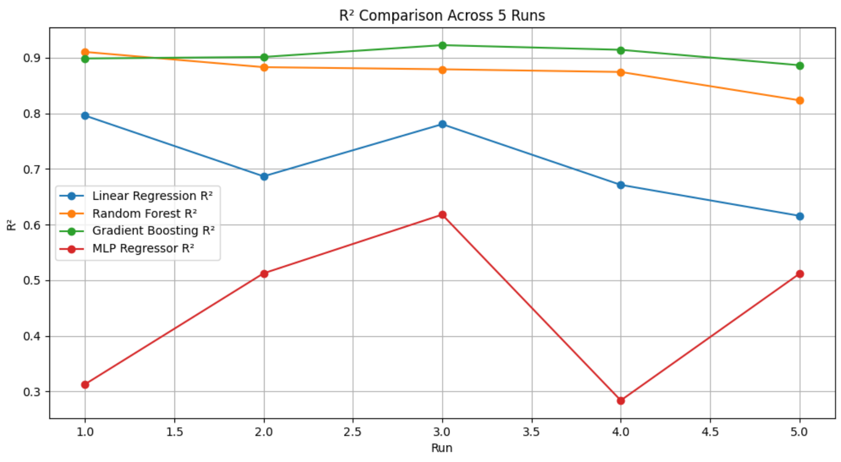
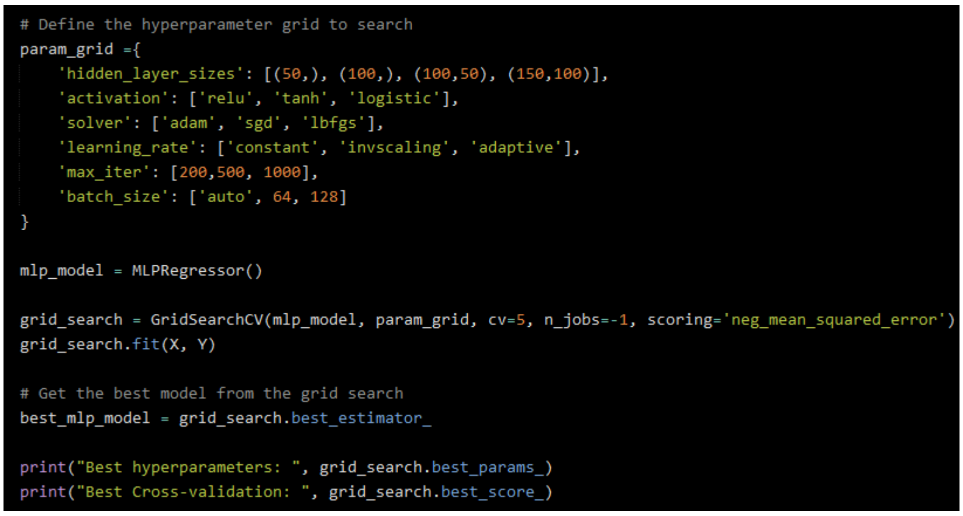


FIGURE 10: R^2 COMPARISON ACROSS 5 RUNS

MLP using hyperparameter tuning

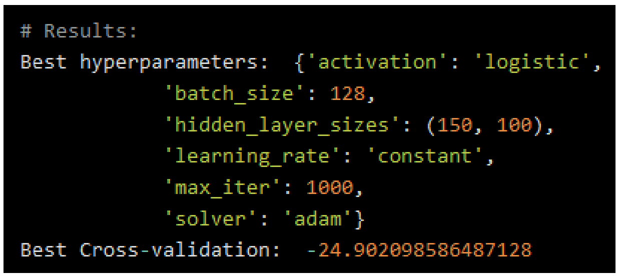
I ran a grid search to determine the best hyperparameter tuning to improve the MLP regressor model. The following code was used in Figure 11:



```
# Define the hyperparameter grid to search
param_grid ={
    'hidden_layer_sizes': [(50,), (100,), (100,50), (150,100)],
    'activation': ['relu', 'tanh', 'logistic'],
    'solver': ['adam', 'sgd', 'lbfgs'],
    'learning_rate': ['constant', 'invscaling', 'adaptive'],
    'max_iter': [200,500, 1000],
    'batch_size': ['auto', 64, 128]
}

mlp_model = MLPRegressor()

grid_search = GridSearchCV(mlp_model, param_grid, cv=5, n_jobs=-1, scoring='neg_mean_squared_error')
grid_search.fit(X, Y)

# Get the best model from the grid search
best_mlp_model = grid_search.best_estimator_

print("Best hyperparameters: ", grid_search.best_params_)
print("Best Cross-validation: ", grid_search.best_score_)
```

FIGURE 11: BEST HYPERPARAMETER GRID SEARCH SETUP

The results that came back where then used to improve the results for the MLP model as the random forest and gradient boosting models were performing well. Figure 12 shows the best parameters.



```
# Results:
Best hyperparameters:  {'activation': 'logistic',
            'batch_size': 128,
            'hidden_layer_sizes': (150, 100),
            'learning_rate': 'constant',
            'max_iter': 1000,
            'solver': 'adam'}
Best Cross-validation:  -24.902098586487128
```

FIGURE 12: BEST PARAMETERS FOR MLP MODEL

After running the models with the new hyper parameter tuning, I obtained the following results shown in Table 3 and Table 4.

TABLE 3: SINGLE RUN RESULTS WITH HYPERPARAMETER TUNING

	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
MAE	3.3605	2.5886	2.2014	2.9676
RMSE	4.7828	3.6941	3.0626	4.2930
R^2	0.7254	0.8362	0.8874	0.7787

TABLE 4: AVERAGE CROSS-VALIDATION RESULTS WITH HYPERPARAMETER TUNING

	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
CV-MAE	3.5999	2.2539	2.1206	3.0394
CV-RMSE	5.3517	3.8588	3.4003	4.5177
CV-R^2	0.7351	0.8568	0.8904	0.7730

From the results, the hyperparameter tuning has had a significant impact on the MLP models values in comparison to the linear regression model. In both the single run and the average results across folds, the MLP outperforms the linear regression model on all metrics. But it still falls behind in comparison to the random forest and gradient boosting models. On average, the MAE of the MLP is still $1000 higher than the random forest and gradient boosting models including having an R^2 value below the 80% target. Below are the graphs showing the difference in results across the 5 runs.

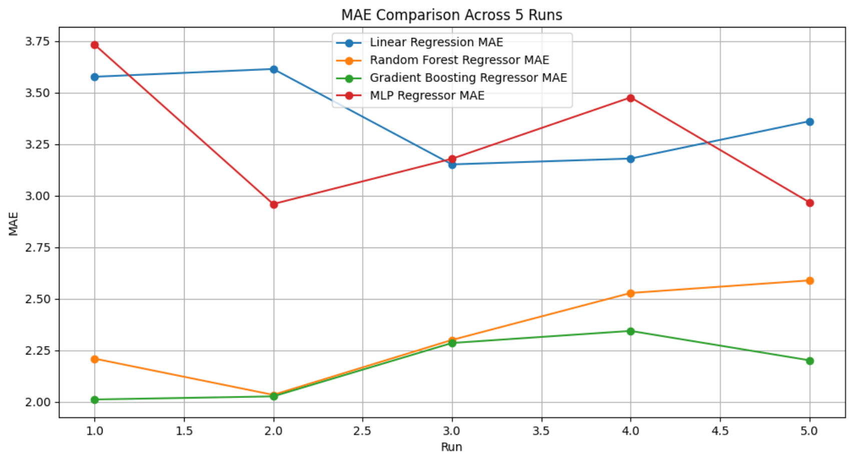


FIGURE 13: MAE BEST PARAMETERS ACROSS 5 RUNS

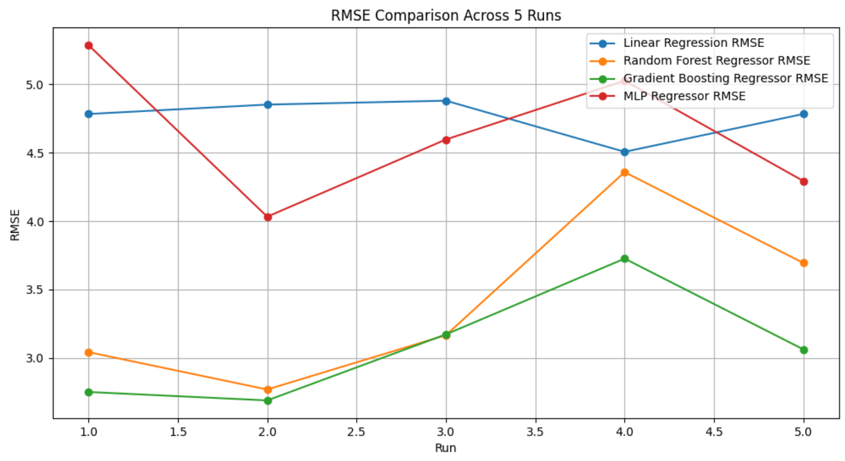


FIGURE 14: RMSE BEST PARAMETERS ACROSS 5 RUNS

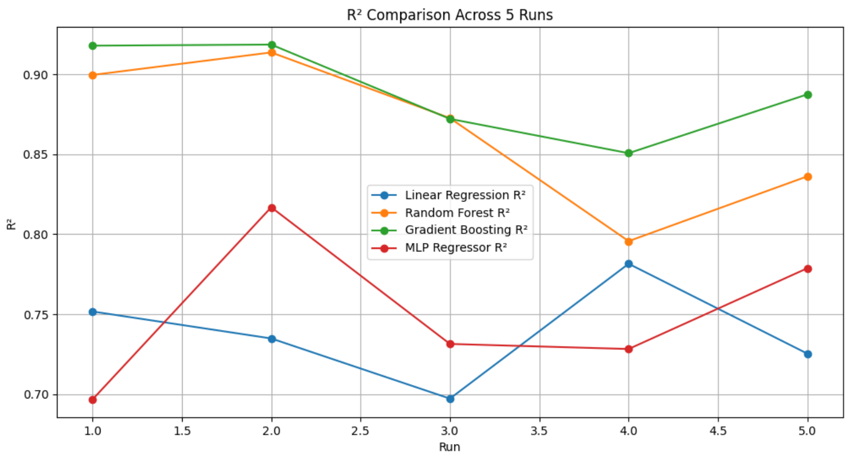
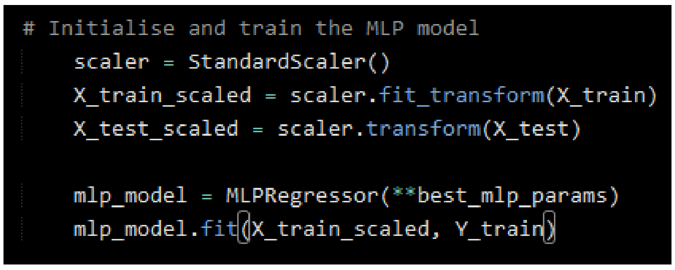


FIGURE 15: R^2 BEST PARAMETERS ACROSS 5 RUNS

MLP using best parameters and standard scaler

Currently, the best model to use has been the gradient boosting model surpassing the random forest as it is able to accurately find relationships in complex and non-linear datasets. The next step is to add a standard scaler to the MLP model which can vastly improve it's results as the model is sensitive to the scale of its input features. Below is the code used to add the standard scaler in Figure 16.



```
# Initialise and train the MLP model
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    mlp_model = MLPRegressor(**best_mlp_params)
    mlp_model.fit(X_train_scaled, Y_train)
```

FIGURE 16: STANDARD SCALER SETUP FOR MLP

With the standard scaler setup, I obtained the following results in Table 5 and Table 6.

TABLE 5: SINGLE RUN RESULTS WITH BEST PARAMETERS AND STANDARD SCALER

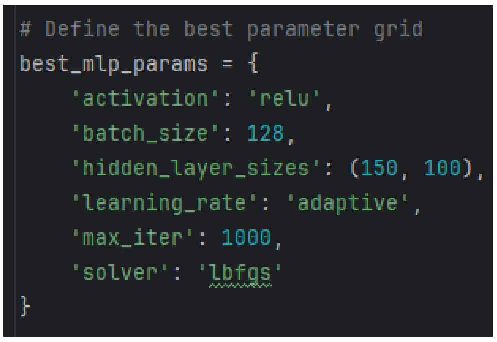
	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
MAE	3.2995	2.0538	1.9735	2.2837
RMSE	4.7515	3.0276	2.7767	3.2916
R^2	0.6976	0.8772	0.8967	0.8549

TABLE 6: AVERAGE RESULTS WITH BEST PARAMETERS AND STANDARD SCALER

	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
CV-MAE	3.5280	2.1014	2.1126	2.7286
CV-RMSE	4.5728	3.0806	2.9618	4.4594
CV-R^2	0.6873	0.8948	0.8878	0.7199

There is a massive improvement in the MLP models' performance in the single run as the R^2 value is 0.8549 meaning it can predict 85.5% of the variance correctly and the RMSE value has been brought down by around $1000. However, the average results across the 5 folds are no better and, in some area, worse than before. The R^2 value has dropped, and the RMSE has increased to a similar value as the linear regressor. I am not happy with these results, so I tweaked the best parameters to try and improve the results.

Figure 17 shows the new best parameters that I have settled on following some trial and error.



```
# Define the best parameter grid
best_mlp_params = {
    'activation': 'relu',
    'batch_size': 128,
    'hidden_layer_sizes': (150, 100),
    'learning_rate': 'adaptive',
    'max_iter': 1000,
    'solver': 'lbfgs'
}
```

FIGURE 17: NEW BEST PARAMETERS

By changing the activator, learning rate and the solver, I was able to improve the results to a more desirable level shown in Table 7 and Table 8.

TABLE 7: SINGLE RUN RESULTS WITH NEW BEST PARAMETERS

	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
MAE	3.5050	1.6760	1.7037	2.5912
RMSE	4.9615	2.3033	2.2503	3.7163
R^2	0.7406	0.9441	0.9466	0.8545

TABLE 8: AVERAGE RESULTS WITH NEW BEST PARAMETERS

	Linear Regression	Random Forest	Gradient Boosting	MLP Regressor
CV-MAE	3.5013	2.1876	2.1107	2.5394
CV-RMSE	5.3797	3.3332	3.1315	4.1856
CV-R^2	0.6869	0.8617	0.8917	0.7953

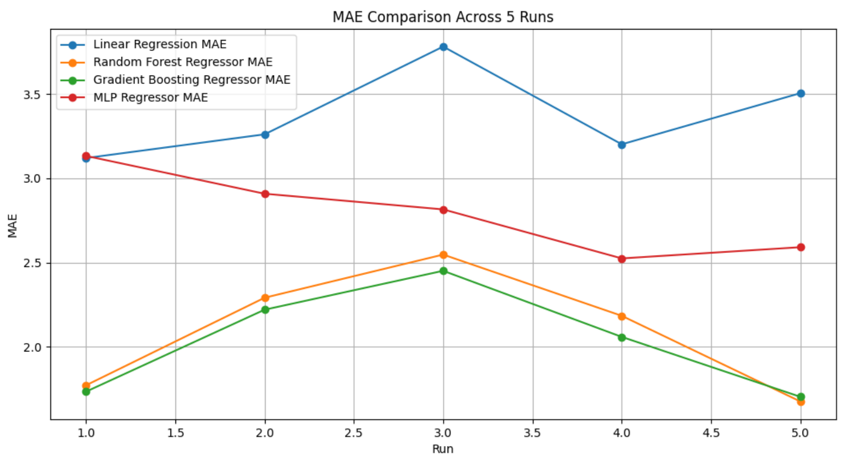


FIGURE 18: MAE WITH NEW BEST PARAMETERS

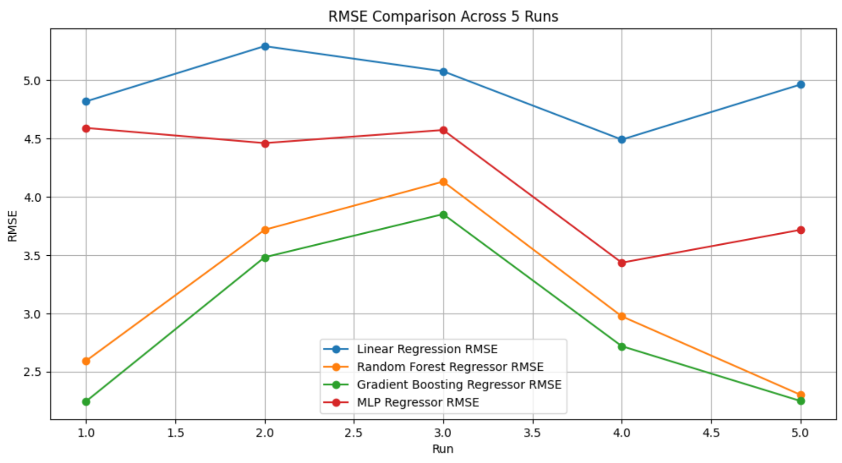


FIGURE 19: RMSE WITH NEW BEST PARAMETERS

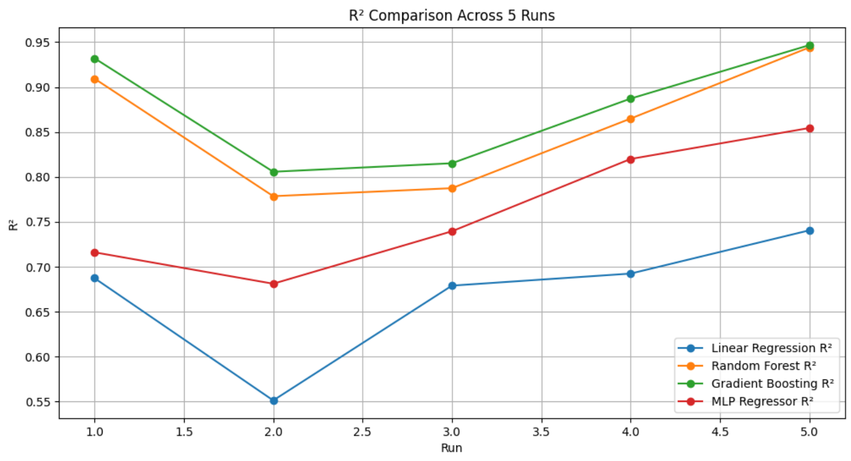


FIGURE 20: R^2 WITH NEW BEST PARAMETERS

The results are still not perfect, and the gradient boosting algorithm is still the clear choice for this dataset, but the MLP algorithm has been vastly improved in comparison to the linear regressor. To do a final comparison between the MLP model and the linear regressor and gradient boost models, the average mean absolute error is lower by 27.5% and higher by 16.9% respectively. For the root mean square error, it is lower by 28.5% and higher by 25.2% respectively. Finally, for the R^2 value, it is higher by 13.6% and lower by 12.1% respectively.

Conclusion

I wanted to get the average cross-fold validation of the MLP model to above 80% and I cannot confidently say that I have achieved that. In a single run this is possible but across 5 different folds the average is not reliably above 80%. To improve this, I would suggest preparing the data more than I have done for this report. An example of this would be to use feature selection and correlation heat maps like the one in figure 6 [3]. Also, using more folds might have an affect on the overall average but it might also have a negative effect. However, the best gradient for this dataset is the gradient boosting regressor, it required no tuning and had the most consistent scores amongst the models.

The dataset presents some ethical and professional issues. The feature column where the number of African Americans is listed should not be considered to affect the price of houses. This only serves to enforce the negative stereotype that the higher the number the worse the area. Another issue is the target column values are misrepresented as properties with a value larger than 50 ($50,000), are not counted in the dataset [14]. This has a detrimental impact on the model as the data is not genuine and therefore, it is difficult to say whether the model is accurate or not.

References

[1] HM Land Registry, “House Price Statistics - UK House Price Index,” *Data.gov.uk*, 2022. https://landregistry.data.gov.uk/app/ukhpi/browse?from=2012-01-01&location=http%3A%2F%2Flandregistry.data.gov.uk%2Fid%2Fregion%2Fbarking-and-dagenham&to=2022-01-01&lang=en (accessed Jan. 10, 2025).

[2] A. Muller, “Data Splitting Strategies — Applied Machine Learning in Python,” *amueller.github.io*, 2020. https://amueller.github.io/aml/04-model-evaluation/1-data-splitting-strategies.html

[3] A. Agarwal, “Linear Regression on Boston Housing Dataset,” *Medium*, Oct. 05, 2018. https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155

[4] K. Grace-Martin, “Overfitting in Regression Models,” *The Analysis Factor*, Aug. 09, 2021. https://www.theanalysisfactor.com/overfitting-regression-models/

[5] D. Nautiyal, “Underfitting and Overfitting in Machine Learning,” *GeeksforGeeks*, Nov. 23, 2017. https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

[6] AnalytixLabs, “Random Forest Regression — How it Helps in Predictive Analytics?,” *Medium*, Dec. 26, 2023. https://medium.com/@byanalytixlabs/random-forest-regression-how-it-helps-in-predictive-analytics-01c31897c1d4

[7] H. Idrees, “Gradient Boosting vs. Random Forest: Which Ensemble Method Should You Use?,” *Medium*, Oct. 14, 2024. https://medium.com/@hassaanidrees7/gradient-boosting-vs-random-forest-which-ensemble-method-should-you-use-9f2ee294d9c6

[8] Z. Zhang, Y. Zhao, A. Canes, D. Steinberg, and O. Lyashevska, “Predictive analytics with gradient boosting in clinical medicine,” *Annals of Translational Medicine*, vol. 7, no. 7, pp. 152–152, Apr. 2019, doi: https://doi.org/10.21037/atm.2019.03.29.

[9] S. Jeske and MarketMuse, “What is a Multilayer Perceptron (MLP) - Multilayer Perceptron (MLP) Definition from MarketMuse Blog,” *MarketMuse Blog*, Mar. 18, 2024. https://blog.marketmuse.com/glossary/multilayer-percpetron-definition/ (accessed Jan. 10, 2025).

[10] Newcastle University, “Coefficient of Determination, R-squared,” *www.ncl.ac.uk*, 2024. https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html (accessed Jan. 10, 2025).

[11] D. Burch, “Mean Absolute Error In Machine Learning: What You Need To Know,” *Arize AI*, Sep. 2023. https://arize.com/blog-course/mean-absolute-error-in-machine-learning-what-you-need-to-know/ (accessed Jan. 10, 2025).

[12] Zach Bobbitt, “MSE vs. RMSE: Which Metric Should You Use?,” *Statology*, Sep. 30, 2021. https://www.statology.org/mse-vs-rmse/ (accessed Jan. 10, 2025).

[13] Raheel Shaikh, “Cross Validation Explained: Evaluating estimator performance.,” *Medium*, Nov. 26, 2018. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85 (accessed Jan. 10, 2025).

[14] D. Harrison and D. L. Rubinfeld, “Boston Dataset,” *www.cs.toronto.edu*, Oct. 10, 1996. https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html (accessed Jan. 11, 2025).