Logo, company name

Description automatically generatedDSA 3362 Final report on Boston Housing Price

**Using unsupervised and supervised method to analyse the Dataset**

Submitted by

Low Eeron (A0216716E)

1. **Introduction**

The Boston Housing dataset offers an opportunity to explore housing prices using machine learning techniques. This study employs supervised learning to predict median home values and unsupervised learning for neighborhood segmentation. Our goal is to provide valuable insights for stakeholders and optimize the property search and marketing processes in the Boston real estate market.

1. **Research Questions**
   1. **Unsupervised Learning Question:**

How can we leverage clustering techniques to segment Boston neighbourhoods based on key characteristics to provide valuable insights to homebuyers and optimize real estate marketing strategies?

* 1. **Supervised Learning Question:**

How effectively can we develop a predictive model to estimate the median housing prices in Boston neighbourhoods, allowing real estate agents to better meet customer preferences and streamline the property search process?

1. **Unsupervised Learning: Clustering Analysis**

**3.1 Kmeans Clustering**

The results from the k-means clustering, when visualized using ggplot, reveal the following relationships between the medv (median value of owner-occupied homes) and the selected variables:

1. For medv against nox (nitric oxide concentration): Group 2 displays a high medv and low nox, while Group 1 shows a low medv and high nox.
2. For medv against indus (proportion of non-retail business acres per town): Group 2 has a high medv and low indus, whereas Group 1 exhibits a low medv and high indus.
3. For medv against rm (average number of rooms per dwelling): Group 2 is characterized by a high medv and high rm, while Group 1 has a low medv and low rm.
4. For medv against lstat (% lower status of the population): Group 2 demonstrates a high medv and low lstat, while Group 1 presents a low medv and high lstat.
5. For medv against ptratio (pupil-teacher ratio by town): Group 2 has a high medv and low ptratio, whereas Group 1 displays a low medv and high ptratio.

These findings highlight the distinct characteristics of the two groups, with Group 2 generally associated with higher median home values and more favorable neighborhood features, while Group 1 shows lower median home values and less desirable neighborhood features

* 1. **Hierarchical Clustering**

The results from the hierarchical clustering using different linkage methods, when analyzed, reveal the following relationships between medv (median value of owner-occupied homes) and the selected variables:

1. When using single linkage, the characteristics of Group 1 and Group 2 differ as follows:
   1. Group 1 exhibits low medv and high nox (nitric oxide concentration), while Group 2 has high medv and high nox.
   2. Group 1 displays low medv and high indus (proportion of non-retail business acres per town), while Group 2 has high medv and low indus.
   3. Group 1 demonstrates low medv and low rm (average number of rooms per dwelling), whereas Group 2 shows high medv and high rm.
   4. Group 1 has low medv and high lstat (% lower status of the population), while Group 2 shows high medv and low lstat.
   5. Group 1 exhibits low medv and high ptratio (pupil-teacher ratio by town), while Group 2 displays high medv and low ptratio.
2. When using complete, average, and Ward's linkage, the characteristics of Group 1 and Group 2 differ as follows:
   1. Group 1 exhibits high medv and low nox (nitric oxide concentration), while Group 2 has low medv and high nox.
   2. Group 1 displays high medv and low indus (proportion of non-retail business acres per town), while Group 2 has low medv and high indus.
   3. Group 1 demonstrates high medv and high rm (average number of rooms per dwelling), whereas Group 2 shows low medv and low rm.
   4. Group 1 has high medv and low lstat (% lower status of the population), while Group 2 shows low medv and high lstat.
   5. Group 1 exhibits high medv and low ptratio (pupil-teacher ratio by town), while Group 2 displays low medv and high ptratio.

These findings highlight the distinct characteristics of the groups in each linkage method. In single linkage, Group 2 generally associates with higher median home values and more favorable neighborhood features, while Group 1 shows lower median home values and less desirable neighborhood features. For complete, average, and Ward's linkage, the roles of Group 1 and Group 2 are reversed, with Group 1 being associated with higher median home values and more favorable neighborhood features, while Group 2 exhibits lower median home values and less desirable neighborhood features.

**3.3 DBSCAN Clustering**

The results from the DBSCAN clustering analysis, when visualized, reveal the following relationships between the medv (median value of owner-occupied homes) and the selected variables:

1. For medv against nox (nitric oxide concentration): Group 2 displays a high medv and low nox, while Group 1 shows a low medv and high nox.
2. For medv against indus (proportion of non-retail business acres per town): Group 2 has a high medv and low indus, whereas Group 1 exhibits a low medv and high indus.
3. For medv against rm (average number of rooms per dwelling): Group 2 is characterized by a high medv and high rm, while Group 1 has a low medv and low rm.
4. For medv against lstat (% lower status of the population): Group 2 demonstrates a high medv and low lstat, while Group 1 presents a low medv and high lstat.
5. For medv against ptratio (pupil-teacher ratio by town): Group 2 has a high medv and low ptratio, whereas Group 1 displays a low medv and high ptratio.

These findings highlight the distinct characteristics of the two groups, with Group 2 generally associated with higher median home values and more favorable neighborhood features, while Group 1 shows lower median home values and less desirable neighborhood features.

**3.4 Comparison of Clustering Techniques**

When comparing the results from k-means clustering, hierarchical clustering, and DBSCAN clustering analyses, the most common relationships between medv and the five selected variables for each group can be summarized as follows:

For Group 1:

1. Low medv and high nox (nitric oxide concentration)
2. Low medv and high indus (proportion of non-retail business acres per town)
3. Low medv and low rm (average number of rooms per dwelling)
4. Low medv and high lstat (% lower status of the population)
5. Low medv and high ptratio (pupil-teacher ratio by town)

For Group 2:

1. High medv and low nox (nitric oxide concentration)
2. High medv and low indus (proportion of non-retail business acres per town)
3. High medv and high rm (average number of rooms per dwelling)
4. High medv and low lstat (% lower status of the population)
5. High medv and low ptratio (pupil-teacher ratio by town)

These common relationships indicate that Group 1 is generally characterized by lower median home values (medv) and less desirable neighborhood features, while Group 2 is associated with higher median home values (medv) and more favorable neighborhood features.

* 1. **Implications for Real Estate Professionals**

The clustering analyses using k-means, hierarchical clustering, and DBSCAN have successfully segmented Boston neighborhoods into two distinct groups based on key characteristics and median home values (medv). This segmentation effectively addresses the question of leveraging clustering techniques to provide valuable insights to homebuyers and optimize real estate marketing strategies.

By identifying the common relationships between medv and the five selected variables (nox, indus, rm, lstat, and ptratio) in each group, real estate professionals can better understand the features that contribute to the desirability and value of neighborhoods. Consequently, they can provide more informed recommendations to homebuyers based on their preferences and budget constraints, streamlining the property search process.

Furthermore, the insights gained from the clustering analysis enable real estate agents to develop targeted marketing strategies for each neighborhood group. For example, Group 1, characterized by lower median home values and less desirable neighborhood features, can be marketed towards budget-conscious buyers or first-time homeowners, while Group 2, associated with higher median home values and more favorable neighborhood features, can be targeted towards buyers seeking upscale properties or prestigious neighborhoods.

In summary, the unsupervised learning techniques have effectively segmented Boston neighborhoods based on key characteristics, enabling real estate professionals to better serve homebuyers and optimize their marketing strategies. By providing insights into the factors that influence housing prices and neighborhood desirability, clustering techniques help enhance the overall efficiency and effectiveness of the real estate market in Boston.

1. **Supervised Learning: Predictive Modeling**

**4.1 Model Evaluation and Selection**

Table1: Overall results of the supervised learning models

|  |  |  |
| --- | --- | --- |
| **Model** | **RMSE** | **MAPE** |
| Decision Tree | 0.5083649 | 1.877817 |
| Bagging | 0.4827810 | 1.491521 |
| Boosting | 0.3769809 | 1.495504 |
| Random Forest | 0.3889372 | 1.207310 |
| Linear Regression | 0.5264129 | 2.290057 |

When choosing the best model for predicting Boston housing prices, it is essential to consider the specific context and the importance of error metrics for the problem. In this case, the Root Mean Square Error (RMSE) is a more relevant metric, as it directly measures the absolute error in housing price predictions, which are continuous numeric values. Among the models evaluated in table 1, the Boosting model has the lowest RMSE (0.3769809), indicating that its predictions are closer to the actual housing prices on average. Therefore, based on the provided results and the nature of the problem, the Boosting model is the recommended choice for predicting Boston housing prices.

**4.2 Application to Real Estate Market**

The Boosting model, which has the lowest RMSE among the models evaluated, provides an effective predictive model for estimating median housing prices in Boston neighborhoods. This model can help real estate agents better understand the relationship between various neighborhood characteristics and housing prices, allowing them to make more accurate predictions about property values.

By developing a predictive model with a low error rate, real estate agents can more effectively match customer preferences with available properties, streamlining the property search process. With accurate estimates of median housing prices, agents can quickly identify neighborhoods that meet clients' budget requirements and desired features, saving both time and effort for all parties involved.

Additionally, the predictive model can serve as a valuable tool for real estate agents to better anticipate market trends and fluctuations in housing prices. Understanding these trends can help agents provide more informed advice to their clients on when to buy or sell properties, maximizing the potential return on investment.

In summary, the Boosting model effectively addresses the topic question by providing an accurate and reliable predictive model for estimating median housing prices in Boston neighborhoods. This, in turn, enables real estate agents to better meet customer preferences and streamline the property search process, ultimately improving the overall efficiency and effectiveness of the real estate market in Boston.

1. **Limitation of Analyses**

The analyses conducted on the Boston housing dataset offer valuable insights into the relationships between various neighborhood characteristics and housing prices. However, there are several limitations that should be acknowledged when interpreting the results and considering their applicability to the current real estate market:

1. Small Sample Size: The dataset contains only 506 samples, which may not be sufficient to capture the full complexity and diversity of Boston's neighborhoods. A small sample size can lead to overfitting in predictive models, limiting their generalizability to other datasets or real-world applications.
2. Outdated Data: The dataset was collected in 1978, and the housing market, neighborhood characteristics, and socioeconomic factors have undoubtedly evolved since then. The results derived from this dataset may not accurately represent the current state of the Boston housing market or the relationships between neighborhood features and housing prices.
3. Limited Variables: The dataset includes only 13 variables, which may not capture all the relevant factors that influence housing prices. Other important factors, such as access to public transportation, walkability, crime rates, and proximity to amenities like parks and shopping centers, are not considered in the analyses. Including additional variables could potentially improve the predictive accuracy of the models and offer a more comprehensive understanding of the factors that contribute to housing prices.
4. Spatial Information: The dataset does not include spatial information, such as neighborhood boundaries or geographic coordinates. This limitation prevents the use of spatial analysis techniques, which could potentially reveal additional insights into the relationships between neighborhood characteristics and housing prices, as well as identify spatial patterns or trends in the data.

In conclusion, while the analyses conducted on the Boston housing dataset provide valuable insights into the relationships between neighborhood features and housing prices, it is essential to recognize the limitations of the dataset and the potential impact on the results' validity and applicability. Future research could benefit from utilizing more recent, comprehensive, and spatially-enabled datasets to better understand and predict housing prices in Boston and other urban areas.

1. **Recommendation for Further Investigation**



Figure1: Process of Data Analytic

Based on the models presented in the analysis, the project has primarily focused on the predictive aspect of analytics, as described by Davenport and Harris (2007) in figure 1. [1] In order to further enhance the value derived from these models, future investigations could explore the development of optimization models for prescriptive analysis, enabling stakeholders to make optimal decisions.

The proposed approach involves utilizing the predictions generated by the current models as inputs for an optimization model, which would be designed to offer prescriptive recommendations for various real estate decisions. These recommendations could then be tested in the real world, allowing for the collection of results and comparison with the model's predictions.

Subsequently, the feedback obtained from these real-world tests can be used to refine and improve the predictive models. This iterative process of refining predictions and optimization can lead to increasingly accurate and effective decision-making tools for the real estate market.

In conclusion, the integration of prescriptive analysis into the current project would create a comprehensive analytics framework, encompassing both predictive and prescriptive components. By continuously refining the models based on real-world feedback, this approach has the potential to significantly enhance the decision-making process for stakeholders in the Boston housing market.

**Reference:**

[1] Davenport, T. H., & Harris, J. G. (2007). Competing on Analytics: The New Science of Winning. Harvard Business Review Press.

**Appendix:**

In the R code, the thought process for selecting the **clustering solution** and **identifying the top 5 variables** can be explained as follows:

Pre-processing the data and determining the optimal number of clusters:

1. The Boston housing price dataset is loaded, and the data is normalized using the **scale()** function.

The within-cluster sum of squares (WCSS) is computed for different numbers of clusters (1 to 10) using k-means clustering. The elbow method is applied to identify the optimal number of clusters, by looking at the first and second differences of WCSS.

1. Evaluating the clustering solution:

Principal Component Analysis (PCA) is performed on the normalized data to reduce its dimensionality and visualize the clustering results more easily.

A scatter plot is created using the first two principal components to examine the clustering results. The plot reveals that the initial optimal number of clusters (9) results in too much overlap and makes it difficult to differentiate between clusters. Therefore, it is decided to use 2 clusters instead.

1. K-means clustering with 2 clusters:

K-means clustering is performed again using 2 clusters, and a new scatter plot is created based on the first two principal components. This plot shows clearer separation between the two clusters.

The cluster centers and summary statistics for each cluster are computed, providing insights into the characteristics of each cluster.

1. Identifying the top 5 variables using logistic regression with LASSO regularization:

A binary target variable is created based on the cluster assignments, and the dataset is split into training and testing sets.

Logistic regression with LASSO regularization is performed to identify the most important variables for predicting cluster membership. LASSO regularization helps in feature selection by shrinking the coefficients of less important variables to zero.

The model's coefficients are extracted, and the top 5 most important variables are identified by sorting the coefficients in descending order of their absolute values.

1. Visualizing the relationship between the top 5 variables and the target variable:

Scatter plots are created for each of the top 5 variables, with the target variable 'medv' (median house value) on the y-axis, and the top 5 variables on the x-axis. This helps to visualize the relationship between these variables and the target variable across the two clusters.

In summary, the final clustering solution consists of two clusters, as it provides better separation and interpretability compared to the initial solution with nine clusters. The top 5 variables are identified using logistic regression with LASSO regularization, which allows for effective feature selection by shrinking the coefficients of less important variables. The scatter plots of these variables against the target variable help to visualize their relationship and relevance to the clusters.

To reach the conclusions described earlier in **kmeans,** the following steps were taken using the R programming language:

1. K-means clustering was performed on the dataset using the **kmeans** function. The dataset was divided into two clusters (Group 1 and Group 2).
2. Cluster assignments were added to the original dataset using the **mutate** function from the **dplyr** package. This created a new dataset called **df\_with\_clusters** that included a new column indicating the cluster membership of each observation.
3. The means of all variables in the dataset were calculated for each cluster using the **group\_by** and **summarize\_all** functions from the **dplyr** package. This resulted in a new dataset called **df\_means**.
4. Scatter plots were created for the top 5 most important variables using **medv** as the y-axis variable. The **ggplot** function from the **ggplot2** package was used to create these plots. The **aes\_string** function was used to specify the x and y variables, as well as the color of the points based on the cluster membership. The **geom\_point** function was used to create the points in the scatter plot, and the **theme\_minimal**, **labs**, and **ggtitle** functions were used to customize the appearance of the plot.
5. The **for** loop iterates through the top 5 most important variables (from **top\_5\_variables$Variable**), generating a scatter plot for each variable against **medv**. The **print(last\_plot())** function inside the loop ensures that each plot is displayed sequentially.

By examining these scatter plots and analyzing the cluster means, conclusions were drawn regarding the relationships between **medv** and the five selected variables for each cluster (Group 1 and Group 2). The distinct characteristics of each group were identified, with Group 1 generally associated with lower median home values and less desirable neighborhood features, while Group 2 was characterized by higher median home values and more favorable neighborhood features.

Here is an explanation of the code and how to perform hierarchical clustering:

1. Load the necessary libraries using the **tidyverse** library.
2. Load the Boston Housing dataset using the **read.csv()** function, and remove the first column using subsetting notation **[, -1]**. Then, normalize the data using the **scale()** function, which scales the data to have a mean of 0 and standard deviation of 1.
3. Define a function called **get\_hclust\_summary()** to perform hierarchical clustering on the normalized data using the Euclidean distance measure and a specified linkage method. The function returns a summary of the resulting clusters by grouping the data by cluster membership and calculating the mean values for each variable.
4. Call the **get\_hclust\_summary()** function for four different linkage methods, "single", "complete", "average", and "ward.D2", and store the resulting summaries in separate data frames.
5. Define a function called **plot\_top\_variables()** to create scatter plots of the top 5 most important variables with medv as the y-axis variable for each of the four linkage methods. The function loops through the top 5 variables and creates a scatter plot for each variable using **ggplot2** and the **geom\_point()** function. The plot's appearance is customized using various **ggplot2** functions and is printed using the **print()** function.
6. Call the **plot\_top\_variables()** function for each of the four linkage methods using the data frames generated in step 4.

To perform hierarchical clustering on your data using R, you can use the **hclust()** function, which performs hierarchical clustering on a distance matrix. The **dist()** function can be used to calculate a distance matrix based on the chosen distance metric. After performing hierarchical clustering, you can use the **cutree()** function to cut the tree at a specified height or number of clusters and obtain the resulting cluster memberships. You can also use the **plot()** function to visualize the resulting dendrogram.

here's an explanation of the code for DBSCAN clustering in a step-by-step format:

1. Load the necessary packages, dbscan and tidyverse, using the library() function. Load the Boston Housing dataset using the read.csv() function, and remove the first column using the subsetting notation [, -1]. Normalize the data using the scale() function, which scales the data to have a mean of 0 and standard deviation of 1.
2. Define a function called plotdist() to generate a k-distance graph for a given dataset and minimum number of points (minPts) parameter. The kNNdist() function from the dbscan package is used to calculate the k-distance for each point, and the resulting distances are sorted in ascending order. The ggplot2 package is then used to generate a scatter plot of the sorted distances, with the x-axis representing the index of each point and the y-axis representing the distance to its kth nearest neighbor.
3. Call the plotdist() function with the normalized data and a chosen minPts value of 15 to generate a k-distance graph. The appropriate eps value can be chosen by examining the graph and looking for the knee point, where the rate of change in the distances starts to level off.
4. Manually set the eps\_value to 5.2 based on the k-distance graph.
5. Use the dbscan() function from the dbscan package to cluster the normalized data using the chosen eps and minPts values. The resulting dbscan\_fit object contains information on the cluster assignments for each observation.
6. Examine the cluster assignments by printing the dbscan\_fit$cluster object.
7. Calculate the means of each variable for each cluster using the group\_by() and summarize\_all() functions from the dplyr package. This results in a summary dataset df\_dbscan\_summary.
8. Add the cluster assignments to the original dataset using the mutate() function and create a new dataset called df\_with\_clusters.
9. Calculate the means of each variable for each cluster using the group\_by() and summarize\_all() functions on df\_with\_clusters, resulting in a summary dataset called df\_means.
10. Use a for loop to create scatter plots for the top 5 most important variables against medv, using the mean values for each variable within each cluster from df\_means. The ggplot2 package is used to generate the plots, with the aes\_string() function used to specify the x and y variables and the color based on the cluster membership. The resulting plots provide insight into the relationship between each variable and the target variable medv for each cluster.

In summary, DBSCAN is a density-based clustering algorithm that partitions the dataset into clusters based on their density. The algorithm requires the specification of two parameters: eps, the maximum distance between two points to be considered in the same cluster, and minPts, the minimum number of points required to form a dense region. The k-distance graph can be used to determine an appropriate value for eps. The resulting clusters can be visualized using scatter plots of the mean values of the variables for each cluster.

Here's an explanation of the code and how to perform regression using decision trees, bagging, boosting, and random forests:

1. Load the necessary libraries using the library() function. The libraries rpart, rpart.plot, caret, tidyverse, Metrics, ipred, and gbm are used for regression using different techniques.
2. Load the Boston Housing dataset using the read.csv() function, and remove the first column using subsetting notation [, -1]. Then, normalize the data using the scale() function, which scales the data to have a mean of 0 and standard deviation of 1. Convert the resulting matrix to a data frame using as.data.frame() function.
3. Split the data into training and testing sets using the sample() function with a seed value of 123. The training set will consist of 80% of the data.
4. Perform decision tree regression using the train() function from the caret package. The medv variable is regressed against all other variables in the training set using the rpart method. The RMSE metric is used to evaluate the model's performance. The resulting model is stored in the tree object. The predict() function is used to generate predictions on the testing set, and the root mean squared error (RMSE) and mean absolute percentage error (MAPE) are calculated using the Metrics package.
5. Perform bagging regression using the train() function from the caret package. The medv variable is regressed against all other variables in the training set using the treebag method. The RMSE metric is used to evaluate the model's performance. The resulting model is stored in the tree\_bagging object. The predict() function is used to generate predictions on the testing set, and the RMSE and MAPE are calculated using the Metrics package.
6. Perform boosting regression using the train() function from the caret package. The medv variable is regressed against all other variables in the training set using the gbm method. The RMSE metric is used to evaluate the model's performance. The resulting model is stored in the tree\_boosting object. The predict() function is used to generate predictions on the testing set, and the RMSE and MAPE are calculated using the Metrics package.
7. Perform random forest regression using the train() function from the caret package. The medv variable is regressed against all other variables in the training set using the rf method. The RMSE metric is used to evaluate the model's performance. The resulting model is stored in the tree\_rf object. The predict() function is used to generate predictions on the testing set, and the RMSE and MAPE are calculated using the Metrics package.

The resulting regression models can be compared based on their performance metrics, and the best model can be used to make predictions on new data.

Below is the explanation of the process for selecting optimal hyperparameter values.

1. Decision Tree:

The optimal cp value was determined by performing a grid search over the range of cp values from 0.001 to 0.1 with 10 equally spaced values.For each cp value, a 5-fold cross-validation was performed, and the average RMSE was computed.The cp value with the lowest average RMSE from the cross-validation was chosen as the optimal value.

1. Bagging:

The optimal nbagg value was determined by performing a grid search over the nbagg values of 10, 25, 50, and 100.For each nbagg value, a 5-fold cross-validation was performed, and the average RMSE was computed.The nbagg value with the lowest average RMSE from the cross-validation was chosen as the optimal value.

1. Boosting:

The optimal combination of interaction.depth, n.trees, shrinkage, and n.minobsinnode was determined by performing a grid search over all possible combinations of their respective values:

interaction.depth: 1, 3, 5, 7, n.trees: 50, 100, 150, shrinkage: 0.1, 0.05, n.minobsinnode: 10, 20

For each combination, a 5-fold cross-validation was performed, and the average RMSE was computed.The combination of hyperparameter values with the lowest average RMSE from the cross-validation was chosen as the optimal combination.

1. Random Forest:

The optimal mtry value was determined by performing a grid search over the mtry values of 2, 4, 6, and 8.For each mtry value, a 5-fold cross-validation was performed, and the average RMSE was computed.The mtry value with the lowest average RMSE from the cross-validation was chosen as the optimal value.

Table2 : summary of the different supervised model and its value

| **Model** | **Hyperparameters** | **Values/Range** | **Cross-validation Folds** | **Selection Criteria** |
| --- | --- | --- | --- | --- |
| Decision Tree | cp (complexity parameter) | 0.001 to 0.1 (10 equally spaced values) | 5 | Lowest average RMSE |
| Bagging | nbagg (number of bagging) | 10, 25, 50, 100 | 5 | Lowest average RMSE |
| Boosting | interaction.depth | 1, 3, 5, 7 | 5 | Lowest average RMSE |
|  | n.trees | 50, 100, 150 |  |  |
|  | shrinkage | 0.1, 0.05 |  |  |
|  | n.minobsinnode | 10, 20 |  |  |
| Random Forest | mtry | 2, 4, 6, 8 | 5 | Lowest average RMSE |

By using grid search and cross-validation, we systematically explored the hyperparameter space for each model and identified the optimal values based on their performance, measured by the average RMSE across the cross-validation folds.