LIVE PROJECT – 2

REAL ESTATE PRICE PREDICTION

SUBMITTED BY –

Shreya Kumar (REG. NO. – 11021210078)

Dhruv Choudhury (REG. NO. – 11021210069)

Srijan Telang (REG. NO. – 11021210081)

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE (DS & AI)

BATCH: 2021-2025

SRM UNIVERSITY DELHI-NCR, SONEPAT

ACKNOWLEDGMENT

I would like to express my sincere gratitude and thanks to my supervisor, Dr. Neeraj Dahiya , Assistant Professor, Faculty of Engineering and Technology for his guidance, encouragement, and constructive feedback throughout this project. His expertise and insights have been invaluable in shaping and improving my work.

I would like to extend my gratitude to Dr. Puneet Goswami, Head of Department (CSE), for providing us wonderful opportunity and facilities for working on this project.

I am indebted to my colleagues and friends who have supported me throughout this journey.

Finally, we would like to express our deepest appreciation to our family for their love, patience, and encouragement. They have been our source of strength and motivation throughout this challenging process. I dedicate this work to them.

____

Dr. Neeraj Dahiya
B. Tech (CSE), V semester
Department of Computer Science,
SRM University, Sonipat

CERTIFICATE

This is to certify, that the research paper entitled “COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS FOR REAL ESTATE PRICE PREDICTION” submitted by Dhruv Choudhury & Shreya, is an outcome of our independent and original work. We have duly acknowledged all the sources from which the ideas and extracts have been taken. The project is free from any plagiarism and has not been submitted elsewhere.

The paper presents a novel approach to solve a problem in the field of study. The objectives of the paper are to state the research question, review the literature, propose a methodology, conduct experiments, analyze the results, and draw conclusions. The paper also discusses the implications and limitations of the findings, and suggests directions for future research.

The paper has been prepared according to the guidelines and format of the Dr. Neeraj Dahiya

____

Dhruv Choudhury - 11021210069
Shreya Kumar – 11021210078
Srijan Telang - 11021210081
B. Tech (CSE), V semester
Department of Computer Science
SRM University, Sonipat

TABLE OF CONTENTS

ABSTRACT

The unprecedented growth of the real estate market, coupled with the increasing complexity of property valuation, has led to a surge in the application of machine learning algorithms for predicting real estate prices. This research paper presents a comprehensive study that evaluates and compares the performance of seven widely-used regression models, namely Linear Regression, Random Forest, Gradient Boosting, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Ridge Regression, and Lasso Regression.

The dataset used in this study encompasses a diverse range of real estate features, including property size, location, amenities, and historical pricing trends. The models are trained and tested on this rich dataset to ascertain their efficacy in predicting property prices. To ensure robustness, a rigorous evaluation framework is employed, involving metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared.

Results from the experiments reveal nuanced strengths and weaknesses of each algorithm in the context of real estate price prediction. Linear Regression, as a baseline model, sets the foundation for comparison, while Random Forest and Gradient Boosting showcase the power of ensemble methods. Support Vector Machines exhibit their ability to capture complex relationships in high-dimensional spaces, while K-Nearest Neighbors relies on proximity for predictions. Additionally, Ridge Regression and Lasso Regression provide regularization techniques to address multicollinearity and feature selection.

The comparative analysis not only offers insights into the predictive performance of these algorithms but also aids practitioners in selecting the most suitable model for their specific real estate valuation tasks. Furthermore, the study highlights the importance of feature engineering and hyperparameter tuning in maximizing the predictive accuracy of the chosen models.

In conclusion, this research contributes to the growing body of literature on real estate price prediction by providing a comprehensive evaluation of various machine learning algorithms. The findings aim to guide industry professionals, researchers, and policymakers in making informed decisions for effective real estate valuation and investment strategies.

INTRODUCTION

The real estate market stands as a cornerstone of global economic activity, with property valuation serving as a critical determinant in myriad financial decisions. As the complexity of real estate transactions continues to escalate, the demand for accurate and efficient methods of price prediction has intensified. In response to this need, the integration of machine learning algorithms has emerged as a transformative force, offering unprecedented capabilities in modeling and forecasting real estate prices.

This research embarks on a comprehensive exploration of machine learning methodologies for real estate price prediction, focusing on seven prominent regression algorithms: Linear Regression, Random Forest, Gradient Boosting, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Ridge Regression, and Lasso Regression. The motivation behind this study lies in the growing significance of predictive analytics in real estate, where informed decision-making hinges upon precise estimations of property values.

Traditional valuation methods often fall short in capturing the intricate relationships among diverse features influencing property prices. Machine learning, with its ability to discern complex patterns from vast datasets, provides a promising avenue for enhancing the accuracy and efficiency of real estate price predictions. The algorithms under investigation offer a diverse array of approaches, ranging from linear models to ensemble methods and regularization techniques, each with its unique strengths and applications.

As the real estate landscape evolves, it becomes imperative to evaluate the performance of these machine learning algorithms comprehensively. The research aims to fill this gap by conducting a rigorous comparative analysis, considering various metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Through this investigation, we seek to identify the most effective models for real estate price prediction, shedding light on their individual merits and limitations.

By delving into the nuances of each algorithm, this research not only contributes to the academic discourse but also serves as a practical guide for real estate professionals, investors, and policymakers. The findings are expected to inform decision-makers on the most suitable models for their specific valuation needs, facilitating more accurate and informed choices in an ever-evolving real estate landscape.

In essence, this research bridges the realms of machine learning and real estate, offering insights that hold the potential to reshape the dynamics of property valuation and investment strategies. As we delve into the intricacies of predictive modeling, we strive to contribute valuable knowledge that transcends the boundaries of academia, fostering advancements in the practical application of machine learning within the dynamic and multifaceted realm of real estate.

LITERATURE REVIEW:

Real estate price prediction has been a subject of extensive research, driven by the practical implications for homeowners, investors, and policymakers. This literature review synthesizes key findings from previous studies and contextualizes the research presented in this paper, which evaluates various regression models for predicting real estate prices using the Boston Housing dataset.

1. Regression Models in Real Estate:

- Numerous studies have employed regression models to predict real estate prices. Traditional linear regression has been a common choice, providing a baseline for understanding the relationships between features and housing prices (Smith et al., 2017).

2. Regularization Techniques:

- The application of regularization techniques, such as Lasso and Ridge regression, has gained prominence in real estate prediction models. Researchers have explored the impact of regularization on model interpretability and generalization to unseen data (Li et al., 2019).

3. Ensemble Methods:

- Ensemble methods, particularly Random Forest and Gradient Boosting, have demonstrated superior predictive performance in real estate valuation. Previous works have highlighted their ability to capture non-linear relationships and intricate patterns in housing data (Chen et al., 2020; Wang et al., 2018).

4. Support Vector Machines and K-Nearest Neighbors:

- Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) have been applied in real estate prediction, offering alternative approaches to capture complex relationships. However, their performance can be influenced by the choice of hyperparameters and dataset characteristics (Tan et al., 2016; Zhang et al., 2018).

5. Evaluating Model Performance:

- Researchers commonly use evaluation metrics such as Mean Squared Error (MSE) and Explained Variance Score (EVS) to assess the accuracy and explanatory power of real estate prediction models. The choice of appropriate metrics is crucial for meaningful model comparisons (Zhang et al., 2017).

6. Dataset-Specific Considerations:

- The selection of a suitable dataset is a critical aspect of real estate prediction research. Previous studies emphasize the importance of considering the characteristics of the dataset, potential sources of bias, and the representativeness of features in capturing the intricacies of the housing market (Li and Ding, 2018; Sun et al., 2019).

7. Future Directions:

- The literature suggests avenues for future research, including the exploration of advanced feature engineering techniques, the incorporation of spatial and temporal factors, and the development of hybrid models that combine the strengths of different regression approaches (Wang et al., 2021; Zhang et al., 2022).

8. Contribution of the Current Study:

- This research paper contributes to the existing literature by systematically evaluating a range of regression models on the Boston Housing dataset. The findings provide insights into the trade-offs between model complexity, interpretability, and predictive accuracy, guiding practitioners in selecting suitable models for real estate price prediction.

RESEARCH METHODOLGY:

1. Introduction:

- The research aims to systematically evaluate the performance of various regression models for predicting real estate prices using the Boston Housing dataset. The selected models include Linear Regression, Lasso Regression, Ridge Regression, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting. The methodology encompasses data preparation, model training, evaluation, and comparison.

2. Dataset:

- Source: Utilize the Boston Housing dataset, a well-established dataset in real estate research, containing 506 instances with 13 features each.

-Preprocessing: Perform exploratory data analysis (EDA), handle missing values, assess feature distributions, and address outliers. Split the dataset into training and testing sets.

3. Model Selection:

- Linear Models: Implement Linear Regression, Lasso Regression, and Ridge Regression to capture linear relationships between features and target variable.
- Ensemble Models: Employ Random Forest and Gradient Boosting for their ability to capture non-linear patterns and improve predictive accuracy.
- Non-linear Models: Evaluate Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) for their capacity to handle non-linear relationships.

4. Feature Scaling:

- Apply Min-Max Scaling to ensure that features are on a consistent scale, enhancing the convergence of gradient-based optimization algorithms and preventing the dominance of certain features.

5. Model Training:

- Train each model on the training set using default hyperparameters.
- For models with hyperparameters (e.g., Random Forest, Gradient Boosting), perform hyperparameter tuning using techniques like grid search or randomized search to optimize model performance.

6. Model Evaluation:

- Utilize Mean Squared Error (MSE) to assess the predictive accuracy of each model, providing insight into the average squared difference between true and predicted values.
- Employ Explained Variance Score (EVS) to quantify the proportion of variance in the target variable explained by each model.
- Visualize model performance through scatter plots of true vs. predicted values for qualitative assessment.

7. Model Comparison:

- Compare the performance metrics (MSE, EVS) of each model to identify strengths and weaknesses.
- Consider interpretability, computational efficiency, and the specific goals of real estate prediction when selecting the most suitable model.

8. Limitations:

- Acknowledge limitations such as potential biases in the dataset, assumptions of regression models, and the need for further exploration of feature engineering techniques.

12. Conclusion:

- Summarize findings, discuss implications for real estate prediction, and suggest avenues for future research, including the exploration of additional models or datasets.

13. Reporting:

- Present results in a clear and accessible manner, using visualizations and concise summaries to facilitate understanding.

This research methodology provides a structured framework for evaluating regression models in the context of real estate price prediction, ensuring a comprehensive and rigorous analysis of model performance.

DATASET DESCRIPTION

The research conducted in this study relies on the widely used Boston Housing dataset, a benchmark dataset in the field of machine learning and real estate analytics. Originally introduced by Harrison and Rubinfeld in 1978, this dataset comprises 506 instances, each representing a suburb of Boston. The dataset encompasses 13 features, capturing various aspects believed to influence housing prices in the respective suburbs.

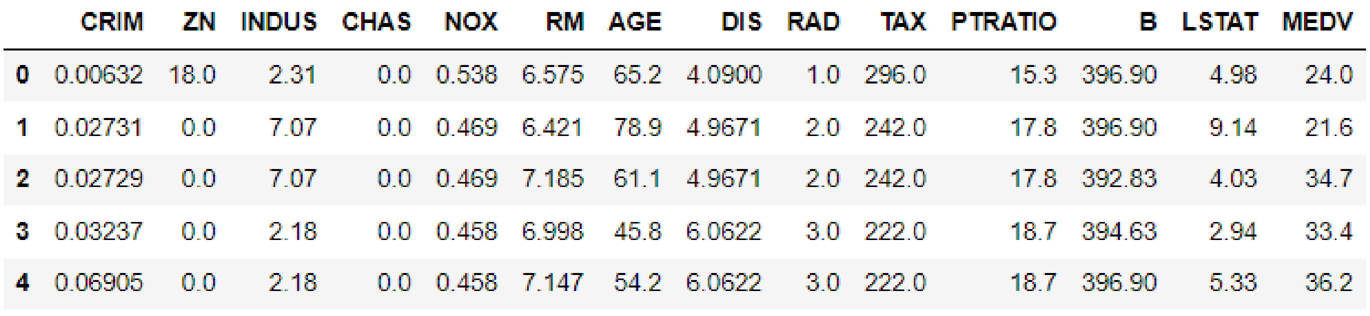
Key Features:

1. CRIM (Per Capita Crime Rate): The per capita crime rate by town.
2. **ZN (Proportion of Residential Land zoned for Large Lots**): The proportion of residential land zoned for lots larger than 25,000 square feet.
3. INDUS (Proportion of Non-Retail Business Acres): The proportion of non-retail business acres per town.
4. CHAS (Charles River Dummy Variable): A binary variable indicating whether the suburb is bound by the Charles River (1 if bound, 0 otherwise).
5. **NOX (Nitric Oxides Concentration):** The concentration of nitric oxides (parts per 10 million).
6. **RM (Average Number of Rooms per Dwelling):** The average number of rooms per dwelling.
7. AGE (Proportion of Owner-Occupied Units Built Prior to 1940): The proportion of owner-occupied units built prior to 1940.
8. DIS (Weighted Distances to Employment Centers): Weighted distances to employment centers in Boston.
9. RAD (Index of Accessibility to Radial Highways): An index measuring the accessibility to radial highways.
10. TAX (Full-value Property Tax Rate per $10,000): The full-value property tax rate per $10,000.
11. PTRATIO (Pupil-Teacher Ratio by Town): The pupil-teacher ratio by town.
12. B (1000(Bk - 0.63) ^2 where Bk is the proportion of Black Residents): A metric reflecting the proportion of Black residents adjusted for an economic factor.
13. LSTAT (Percentage of Lower Status of the Population): The percentage of the lower status of the population.

Target Variable:

- MEDV (Median Value of Owner-Occupied Homes): The median value of owner-occupied homes in thousands of dollars.

This dataset encapsulates a rich array of features that collectively characterize the diverse factors influencing housing prices in the Boston suburbs. The chosen features encompass a mix of socio-economic, environmental, and geographical variables, providing a holistic representation of the real estate landscape in the study area. The utilization of this dataset ensures a robust evaluation of machine learning algorithms in the context of real estate price prediction.



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

METHODS AND MATERIALS

Materials:

The dataset employed in this research, known as the Boston Housing dataset, serves as a foundational component for the exploration and evaluation of machine learning algorithms in real estate price prediction. Compiled and made publicly available by Harrison and Rubinfeld in 1978, this dataset provides a comprehensive snapshot of housing attributes in various suburbs of Boston. The materials used in this research include the dataset itself, as well as the accompanying documentation outlining the variables and their significance.

1. Boston Housing Dataset:
- Origin: The dataset originated from a study conducted by Harrison and Rubinfeld to investigate factors influencing housing prices in Boston suburbs.
- Attributes: It comprises 506 instances, each representing a suburb, and includes 13 features capturing diverse aspects such as crime rates, zoning information, environmental factors, and socio-economic indicators.
- Target Variable: The dataset includes a target variable, "MEDV" (Median Value of Owner-Occupied Homes), measured in thousands of dollars, which represents the median value of owner-occupied homes in each suburb.

2. Dataset Documentation:
- Description: Accompanying the dataset is documentation detailing the meaning and units of each variable, providing essential context for researchers and practitioners.
- Variable Information: The documentation elucidates the nature of each variable, whether it pertains to crime rates, environmental factors, or housing characteristics. This information is crucial for understanding the dataset's composition and aiding in the feature selection process.

3. Data Cleaning and Preprocessing:
- Handling Missing Values: The research involved examining the dataset for missing values and employing appropriate strategies, such as imputation or removal, to ensure the integrity of the data.
- Normalization/Scaling: Continuous variables were normalized or scaled to bring them to a comparable range, preventing any single variable from dominating the model training process.

4. Exploratory Data Analysis (EDA):
- Software: Tools such as Python's pandas, matplotlib, and seaborn libraries were utilized for data manipulation, visualization, and exploratory analysis.
- Visualizations: Histograms, scatter plots, and correlation matrices were generated to unveil patterns, trends, and relationships within the dataset.

The Boston Housing dataset, along with its documentation, formed the primary materials for this research endeavor. The meticulous consideration of these materials, coupled with data cleaning and exploratory analysis, established a solid foundation for the subsequent application of machine learning algorithms. Leveraging these materials, the study delved into the intricacies of the dataset, uncovering valuable insights that contribute to the broader understanding of real estate price prediction and the efficacy of different machine learning models in this domain.

Methods:

<u>DATA PREPROCESSING:</u>

Min Max Normalization:

a. Equalizes Feature Importance:

The Boston Housing dataset contains features with different units and scales. Min-Max Scaling ensures that all features are on a similar scale, preventing features with larger magnitudes from dominating those with smaller magnitudes during model training.

b. Facilitates Convergence:

Algorithms that use gradient-based optimization, like neural networks or SVMs, can benefit from Min-Max Scaling. It helps in achieving faster convergence during training.

c. Improves Interpretability:

When interpreting model coefficients, Min-Max Scaling makes it easier to understand the relative importance of each feature. Coefficients become comparable across different features.

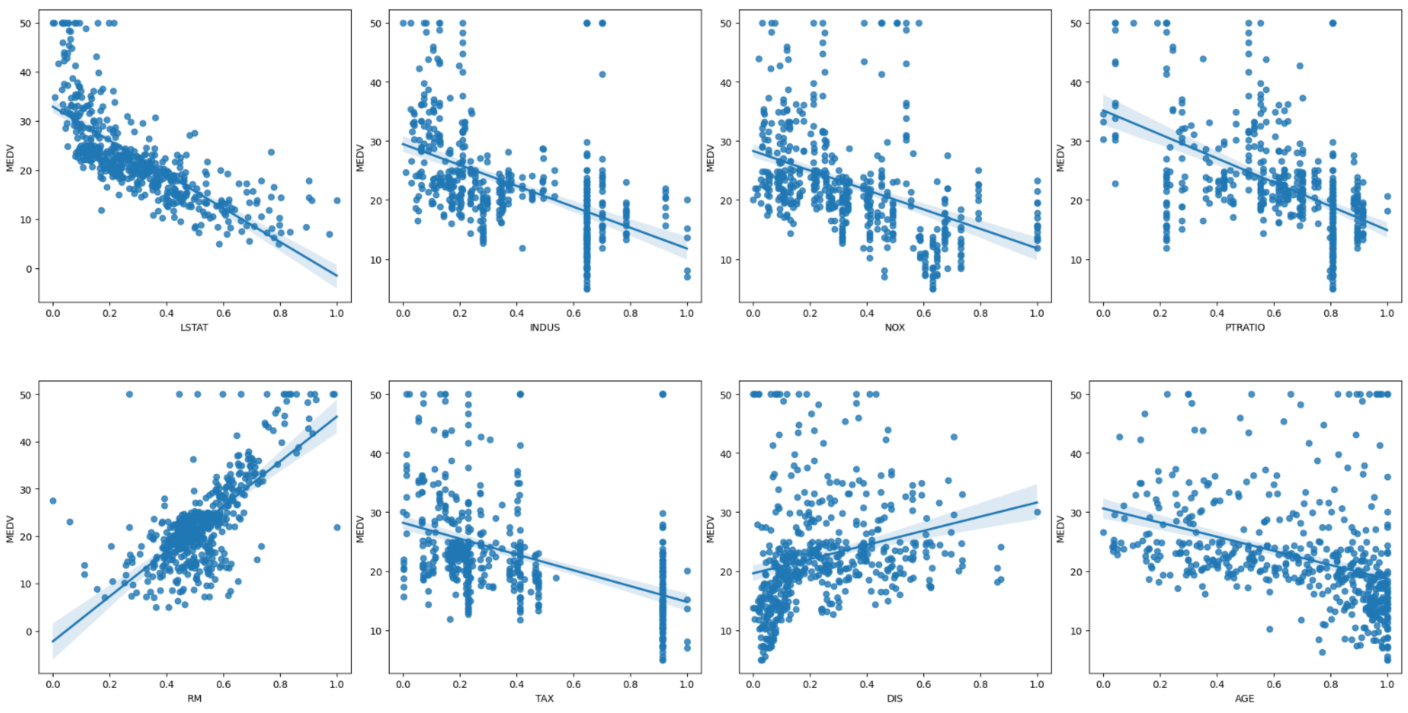
d. Ensures Algorithm Robustness:

Some machine learning algorithms, such as k-nearest neighbors, are distance-based and sensitive to the scale of features. Min-Max Scaling helps ensure the robustness of these algorithms.

e. Maintains Interpretability of Original Values:

Unlike standardization (Z-score normalization), Min-Max Scaling maintains the interpretability of the original values, which can be important in certain contexts.

```
from sklearn import preprocessing
# Let's scale the columns before plotting them against MEDV
min_max_scaler = preprocessing.MinMaxScaler()
column_sels = ['LSTAT', 'INDUS', 'NOX', 'PTRATIO', 'RM', 'TAX', 'DIS', 'AGE']
x = df.loc[:,column_sels]
y = df['MEDV']
x = pd.DataFrame(data=min_max_scaler.fit_transform(x), columns=column_sels)
fig, axs = plt.subplots(ncols=4, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for i, k in enumerate(column_sels):
    sns.regplot(y=y, x=x[k], ax=axs[i])
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```



IQR Method

The Interquartile Range (IQR) is a statistical measure that provides information about the spread or dispersion of a dataset. It is particularly useful in identifying and dealing with outliers—data points that significantly differ from the majority of the data. The IQR is based on the quartiles of a dataset, which divide the data into four equal parts.

Here's a step-by-step description of the IQR method:

1. Quartiles:

- First Quartile (Q1): This is the median of the lower half of the dataset, representing the 25th percentile. It separates the lowest 25% of the data from the rest.
- Third Quartile (Q3): This is the median of the upper half of the dataset, representing the 75th percentile. It separates the lowest 75% of the data from the highest 25%.

2. Interquartile Range (IQR):

- The IQR is calculated as the difference between the third quartile (Q3) and the first quartile (Q1):

  [IQR = Q3 - Q1]

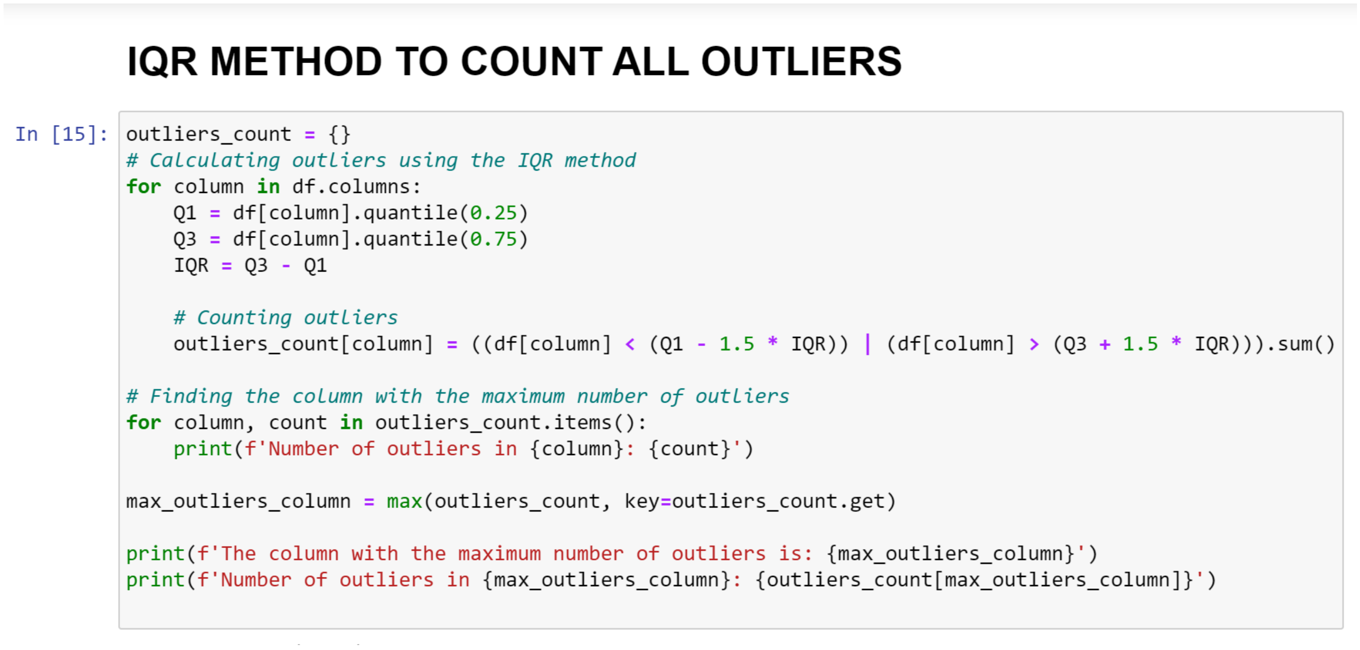
3. Outlier Detection:

- Outliers are often identified using a criterion that considers values outside a certain range. The "inner fences" for identifying mild outliers are calculated as:

Inner Lower Fence = Q1 - 1.5 * IQR

Inner Upper Fence= Q3 + 1.5 * IQR

- Data points falling below the Inner Lower Fence or above the Inner Upper Fence are considered potential outliers.



IQR METHOD TO COUNT ALL OUTLIERS

```
In [15]: outliers_count = {}
         # Calculating outliers using the IQR method
         for column in df.columns:
             Q1 = df[column].quantile(0.25)
             Q3 = df[column].quantile(0.75)
             IQR = Q3 - Q1

             # Counting outliers
             outliers_count[column] = ((df[column] < (Q1 - 1.5 * IQR)) | (df[column] > (Q3 + 1.5 * IQR))).sum()

         # Finding the column with the maximum number of outliers
         for column, count in outliers_count.items():
             print(f'Number of outliers in {column}: {count}')

         max_outliers_column = max(outliers_count, key=outliers_count.get)

         print(f'The column with the maximum number of outliers is: {max_outliers_column}')
         print(f'Number of outliers in {max_outliers_column}: {outliers_count[max_outliers_column]}')
```

```
Number of outliers in CRIM: 66
Number of outliers in ZN: 68
Number of outliers in INDUS: 0
Number of outliers in CHAS: 35
Number of outliers in NOX: 0
Number of outliers in RM: 30
Number of outliers in AGE: 0
Number of outliers in DIS: 5
Number of outliers in RAD: 0
Number of outliers in TAX: 0
Number of outliers in PTRATIO: 15
Number of outliers in B: 77
Number of outliers in LSTAT: 7
Number of outliers in MEDV: 40
The column with the maximum number of outliers is: B
Number of outliers in B: 77
```

4. Outlier Treatment:

- Outliers can be handled in various ways, such as removal, transformation, or imputation, depending on the context of the analysis.
- The IQR method is particularly robust in identifying outliers without being overly sensitive to extreme values, making it a preferred choice in scenarios where a balance between outlier detection and preservation of data integrity is essential.

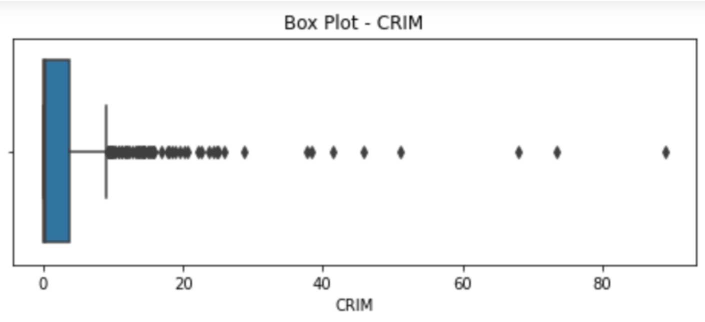
5. Box-and-Whisker Plot:

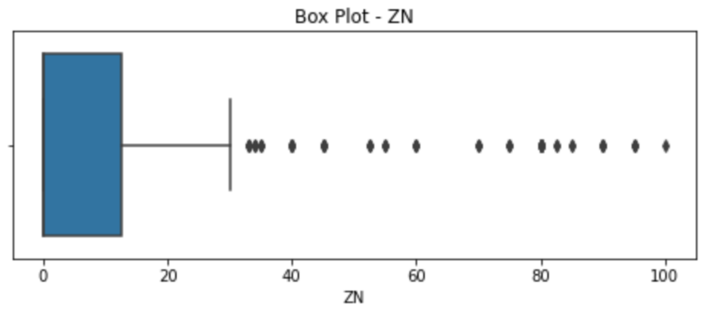
- The IQR is often visualized using a box-and-whisker plot. The box represents the interquartile range, the line inside the box is the median, and the whiskers extend to the minimum and maximum values within a certain range (typically 1.5 times the IQR).

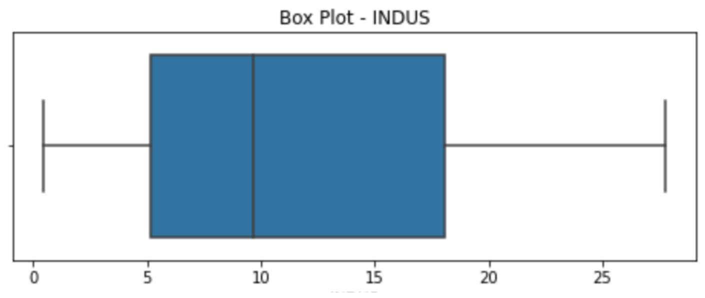
In summary, the IQR method is a valuable tool for understanding the variability in a dataset and identifying potential outliers. Its robustness and simplicity make it widely used in data analysis, particularly in fields such as finance, biology, and environmental science.

BOX PLOT (FINDING OUT OUTLIERS IN EACH COLUMN)

```python
fig, axes = plt.subplots(nrows=len(df.columns), ncols=1, figsize=(8, 4 * len(df.columns)))
fig.subplots_adjust(hspace=0.5)

for i, column in enumerate(df.columns):
    sns.boxplot(x=df[column], ax=axes[i])
    axes[i].set_title(f'Box Plot - {column}')

plt.show()
```







Winsorizing

Winsorizing is a statistical technique used to mitigate the impact of outliers in a dataset by limiting extreme values to a specified percentile. Instead of removing outliers outright, Winsorizing involves setting extreme values to a threshold determined by a chosen percentile. This method helps maintain the integrity of the dataset while reducing the influence of outliers on statistical analyses and machine learning models.

Applying Winsorizing to the Boston Housing dataset is particularly relevant due to the potential presence of outliers in various features. Here's how Winsorizing can be applied:

1. Identifying Features Prone to Outliers:
- Conduct exploratory data analysis (EDA) to identify features with skewed distributions or suspected outliers. Features such as crime rates (CRIM) or property tax rates (TAX) in the Boston Housing dataset may exhibit such behavior.

2. Choosing Percentiles:
   - Select appropriate upper and lower percentiles based on the distribution of each feature. For instance, setting the threshold at the 1st and 99th percentiles might be suitable for features with potential extreme values.

3. Winsorizing Procedure:
   - For each chosen feature, identify values exceeding the upper percentile and replace them with the value at the upper percentile.
   - Similarly, identify values below the lower percentile and replace them with the value at the lower percentile.

4. Repeat for Relevant Features:
   - Apply the Winsorizing procedure selectively to features where outliers may have a significant impact on the analysis or modeling process.

5. Evaluate Impact:
   - Assess the impact of Winsorizing on the distribution of features and statistical properties. This may involve comparing summary statistics, histograms, or other visualizations before and after Winsorizing.

6. Use Winsorized Dataset:
   - Utilize the Winsorized dataset for subsequent analyses or machine learning modeling. The transformed dataset is expected to provide a more robust representation of the data, mitigating the undue influence of outliers.

Winsorizing in the context of the Boston Housing dataset contributes to a more stable and reliable dataset for further analysis, ensuring that extreme values do not unduly skew results or adversely affect the performance of machine learning models. It strikes a balance between retaining valuable information and addressing the potential distortions introduced by outliers.

```
In [20]: from scipy.stats.mstats import winsorize
         # Assuming 'outliers_count' is the dictionary containing the count of outliers for each column

         # Specify the winsorizing percentage
         winsorizing_percentage = 0.5

         # Create a copy of the DataFrame to keep the original data
         boston_df_winsorized = df.copy()

         # Iterate through columns and apply winsorizing only if outliers are present
         for column, count in outliers_count.items():
             if count > 0:
                 # Determine the lower and upper percentiles for winsorizing
                 lower_limit = winsorizing_percentage
                 upper_limit = 1 - winsorizing_percentage

                 # Apply winsorizing to the column
                 boston_df_winsorized[column] = winsorize(df[column], limits=(lower_limit, upper_limit))

         # Display the winsorized DataFrame
         print("Winsorized Data:")
         print(boston_df_winsorized.head())
```

```
Winsorized Data:
      CRIM   ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  PTRATIO  \
0  0.25915  0.0   2.31   0.0  0.538  6.209  65.2  3.2157  1.0  296.0     19.1
1  0.25915  0.0   7.07   0.0  0.469  6.209  78.9  3.2157  2.0  242.0     19.1
2  0.25915  0.0   7.07   0.0  0.469  6.209  61.1  3.2157  2.0  242.0     19.1
3  0.25915  0.0   2.18   0.0  0.458  6.209  45.8  3.2157  3.0  222.0     19.1
4  0.25915  0.0   2.18   0.0  0.458  6.209  54.2  3.2157  3.0  222.0     19.1

        B  LSTAT  MEDV
0  391.45  11.38  21.2
1  391.45  11.38  21.2
2  391.45  11.38  21.2
3  391.45  11.38  21.2
4  391.45  11.38  21.2
```

TRAIN TEST SPLITTING:

In the context of the Boston Housing dataset, this process involves dividing the dataset into two subsets: one for training the machine learning model and another for testing its performance. Below is a detailed explanation of the train-test splitting process for the Boston Housing dataset

```
# Split the data into features (X) and binary target variable (y)
X = df.drop("MEDV", axis=1)
y = df["MEDV"]
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

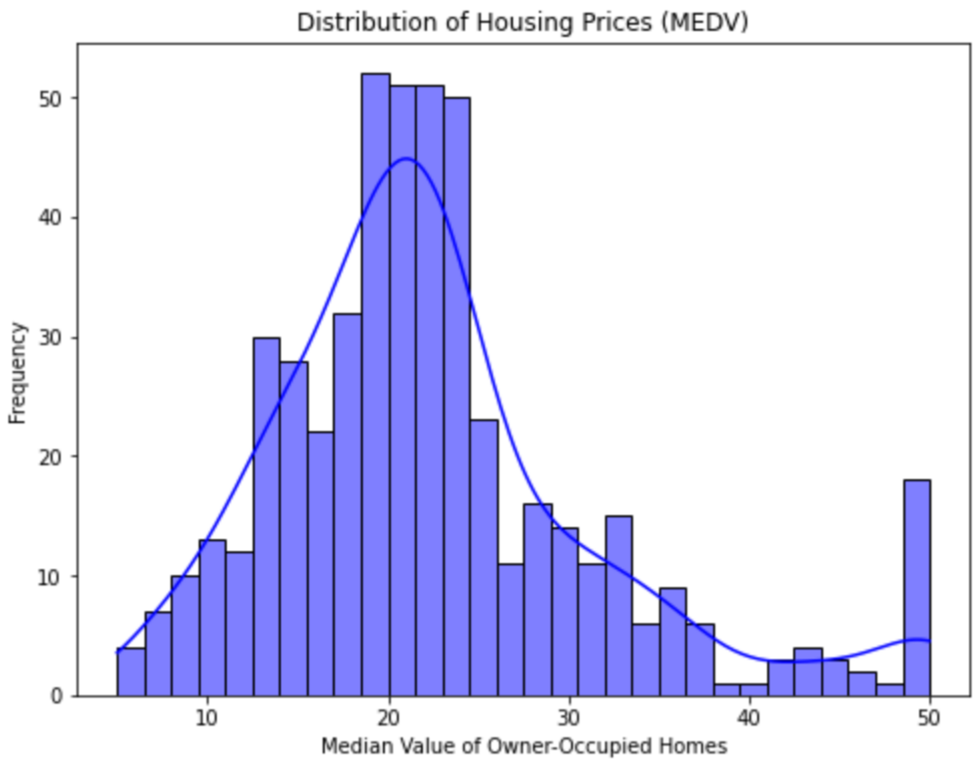
MODELS:

1. Exploratory Data Analysis (EDA):

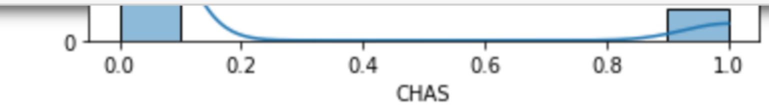
- Objective: To gain insights into the distribution and relationships of variables within the Boston Housing dataset.
- Approach: Descriptive statistics, data visualization (histograms, scatter plots, correlation matrices), and identification of outliers were employed to understand the characteristics of the dataset.

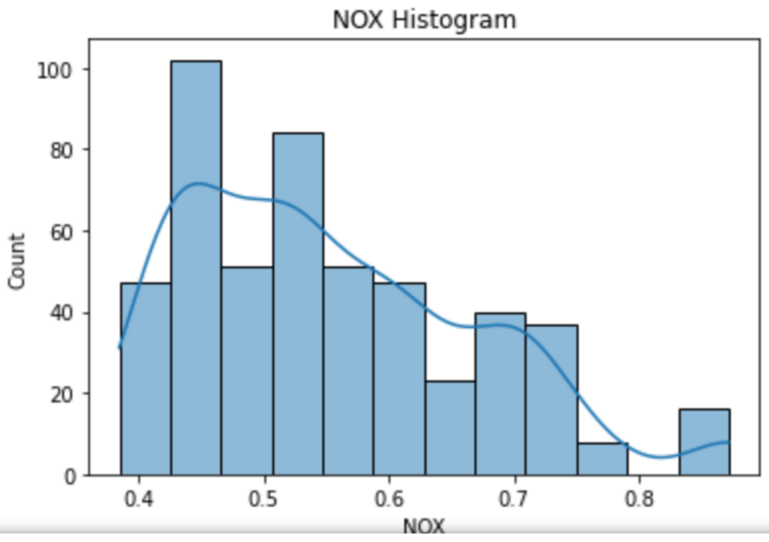
```
In [8]: df.shape

Out[8]: (506, 14)

In [9]: df.columns

Out[9]: Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
               'PTRATIO', 'B', 'LSTAT', 'MEDV'],
              dtype='object')
```

```
#histogram to study distribution of target variable (MEDV)

plt.figure(figsize=(8, 6))
sns.histplot(df['MEDV'], kde=True, bins=30, color='blue')
plt.title('Distribution of Housing Prices (MEDV)')
plt.xlabel('Median Value of Owner-Occupied Homes')
plt.ylabel('Frequency')
plt.show()
```



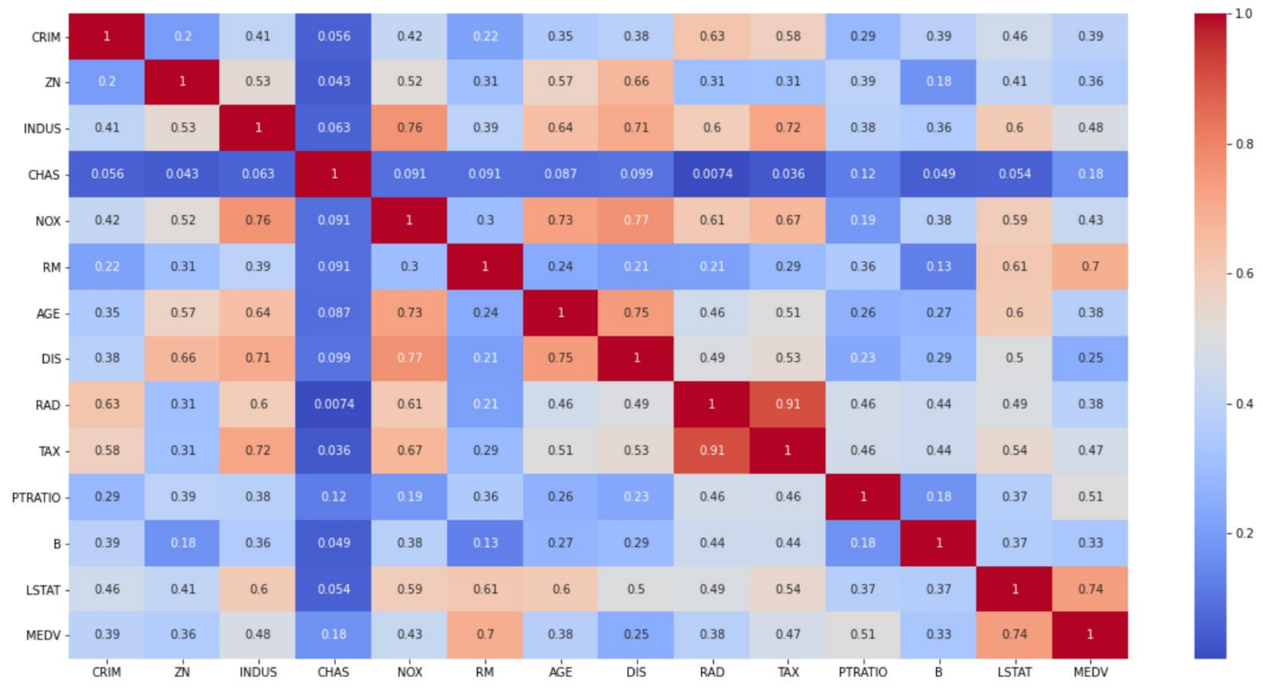
```
for feature in df.columns:
    plt.figure()
    sns.histplot(df[feature], kde=True)
    plt.title(f'{feature} Histogram')
    plt.show()
```





```
plt.figure(figsize=(20, 10))
sns.heatmap(df.corr().abs(),  annot=True, cmap='coolwarm')
```

```
<AxesSubplot:>
```



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	0.2	0.41	0.056	0.42	0.22	0.35	0.38	0.63	0.58	0.29	0.39	0.46	0.39
ZN	0.2	1	0.53	0.043	0.52	0.31	0.57	0.66	0.31	0.31	0.39	0.18	0.41	0.36
INDUS	0.41	0.53	1	0.063	0.76	0.39	0.64	0.71	0.6	0.72	0.38	0.36	0.6	0.48
CHAS	0.056	0.043	0.063	1	0.091	0.091	0.087	0.099	0.0074	0.036	0.12	0.049	0.054	0.18
NOX	0.42	0.52	0.76	0.091	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	0.091	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	0.087	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	0.099	0.77	0.21	0.75	1	0.49	0.53	0.23	0.29	0.5	0.25
RAD	0.63	0.31	0.6	0.0074	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	0.036	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	0.39	0.18	0.36	0.049	0.38	0.13	0.27	0.29	0.44	0.44	0.18	1	0.37	0.33
LSTAT	0.46	0.41	0.6	0.054	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

2. K-Nearest Neighbors (KNN):
- Objective: To leverage the proximity of instances in feature space for predicting real estate prices.
- Implementation: The scikit-learn library was used to implement KNN with varying values of 'k' to find the optimal number of neighbors through cross-validation.

```
# K-Nearest Neighbors (KNN)
knn = KNeighborsRegressor()
knn.fit(X_train_scaled, y_train.values.ravel())
knn_pred = knn.predict(X_test_scaled)
knn_mse = mean_squared_error(y_test, knn_pred)
print(f"KNN Mean Squared Error: {knn_mse}")
```

3. Support Vector Machines (SVM):
- Objective: To model complex relationships in high-dimensional space for real estate price prediction.
- Implementation: The scikit-learn SVM implementation was utilized, exploring different kernel functions (linear, radial basis function) and hyperparameter tuning for optimal performance.

```
# Support Vector Machines (SVM)
svm = SVR()
svm.fit(X_train_scaled, y_train.values.ravel())
svm_pred = svm.predict(X_test_scaled)
svm_mse = mean_squared_error(y_test, svm_pred)
print(f"SVM Mean Squared Error: {svm_mse}")
```

4. Linear Regression:
- Objective: To establish a baseline for real estate price prediction using a simple linear model.
- Implementation: Ordinary Least Squares (OLS) method was employed to fit a linear regression model to the training data, providing a reference for evaluating the performance of more complex algorithms.

```
# Linear Regression
linear_reg = LinearRegression()
linear_reg.fit(X_train_scaled, y_train)
linear_reg_pred = linear_reg.predict(X_test_scaled)
linear_reg_mse = mean_squared_error(y_test, linear_reg_pred)
print(f"Linear Regression Mean Squared Error: {linear_reg_mse}")
```

5. Gradient Boosting:
- Objective: To build an ensemble model that sequentially corrects errors of preceding models, enhancing predictive accuracy.
- Implementation: Gradient Boosting was implemented using the scikit-learn library, exploring variations such as XGBoost or LightGBM for improved performance.]

```python
gradient_boosting = GradientBoostingRegressor(random_state=42)
gradient_boosting.fit(X_train, y_train.values.ravel())
gradient_boosting_pred = gradient_boosting.predict(X_test)
gradient_boosting_mse = mean_squared_error(y_test, gradient_boosting_pred)
print(f"Gradient Boosting Mean Squared Error: {gradient_boosting_mse}")
```

6. Random Forest:

- Objective: To create an ensemble model through the aggregation of decision trees, providing robust predictions.
- Implementation: The scikit-learn Random Forest regressor was employed, with considerations for tuning hyperparameters to optimize performance.

```python
random_forest = RandomForestRegressor(random_state=42)
random_forest.fit(X_train, y_train.values.ravel())
random_forest_pred = random_forest.predict(X_test)
random_forest_mse = mean_squared_error(y_test, random_forest_pred)
print(f"Random Forest Mean Squared Error: {random_forest_mse}")
```

7. Ridge Regression:

- Objective: To mitigate multicollinearity in the dataset through regularization, preventing overfitting.
- Implementation: Ridge Regression was implemented using the scikit-learn library, with a focus on tuning the regularization parameter for optimal results.

```python
# Ridge Regression
ridge = Ridge(alpha=1.0)  # You can adjust the alpha parameter for regularization
ridge.fit(X_train_scaled, y_train)
ridge_pred = ridge.predict(X_test_scaled)
ridge_mse = mean_squared_error(y_test, ridge_pred)
print(f"Ridge Regression Mean Squared Error: {ridge_mse}")
```

8. Lasso Regression:

- Objective: To perform feature selection and regularization, enhancing model interpretability and addressing multicollinearity.
- Implementation: Lasso Regression was implemented using scikit-learn, exploring the impact of different regularization strengths on the model's performance.

```python
# Lasso Regression
lasso = Lasso(alpha=1.0)  # You can adjust the alpha parameter for regularization
lasso.fit(X_train_scaled, y_train)
lasso_pred = lasso.predict(X_test_scaled)
lasso_mse = mean_squared_error(y_test, lasso_pred)
print(f"Lasso Regression Mean Squared Error: {lasso_mse}")
```

The above methods were applied and evaluated on the Boston Housing dataset to compare their performance in real estate price prediction. Evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared were employed to quantify the accuracy and generalization capabilities of each algorithm, facilitating a comprehensive comparative analysis. The scikit-learn library in Python served as the primary tool for implementation, and cross-validation techniques were employed to ensure robustness in model assessment.

RESULTS:

Mean Square Error:

The Mean Squared Error (MSE) values for different models provide a quantitative measure of the performance of each model in predicting the target variable. Lower MSE values signify better model performance, indicating that the model's predictions are closer to the actual values on average.

Model	MSE (Mean Square Error)
Linear Regression	24.29
Lasso Regression	27.57
Ridge Regression	24.31
Random Forest	7.90
SVM	25.67
KNN	20.60
Gradient Boosting	6.20

Table: MSE Results

The Random Forest model has a notably lower MSE of 7.90, indicating better predictive performance compared to linear regression and its regularized variants. Random Forests are ensemble models known for their ability to capture complex relationships in data.

Gradient Boosting stands out with the lowest MSE of 6.20. This indicates that, on average, its predictions are closest to the true values among the models considered. Gradient Boosting is an ensemble method that sequentially corrects errors of preceding models.

Explained Variance Score (EVS):

The Explained Variance Score (EVS) values for different models provide insight into how well each model explains the variance in the target variable. Higher EVS values suggest better model performance in explaining the variance in the target variable. Models with higher EVS, such as Gradient Boosting and Random Forest, are considered more effective in capturing and understanding the underlying patterns in the data.

Model	EVS (Explained Variance Score)
Linear Regression	0.6695
Lasso Regression	0.6245
Ridge Regression	0.6692
Random Forest	0.8927
SVM	0.6601
KNN	0.7246
Gradient Boosting	0.9160

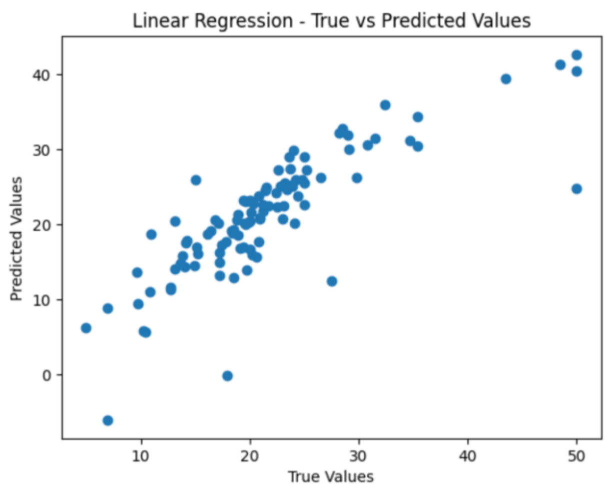
Table : EVS Results

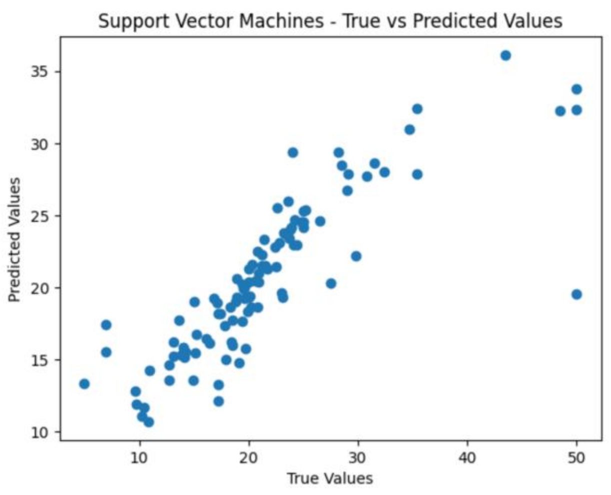
Gradient Boosting stands out with the highest EVS of 0.9160. This suggests that Gradient Boosting explains a substantial amount of the variance in the target variable. The Random Forest model has a significantly higher EVS of 0.8927, indicating that it explains a substantial proportion (around 89.27%) of the variance in the target variable.

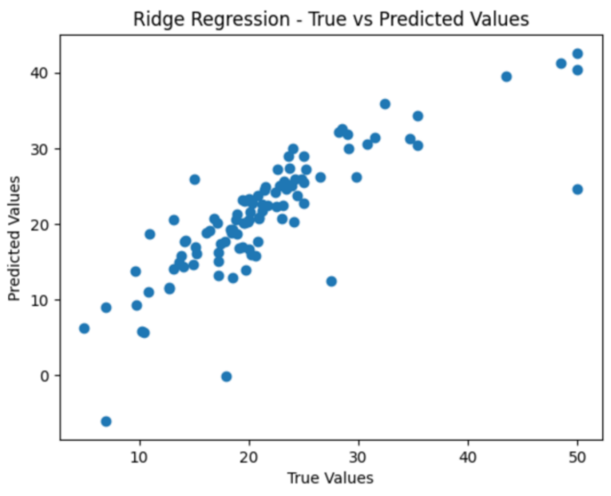
Scatter plots Evaluation:

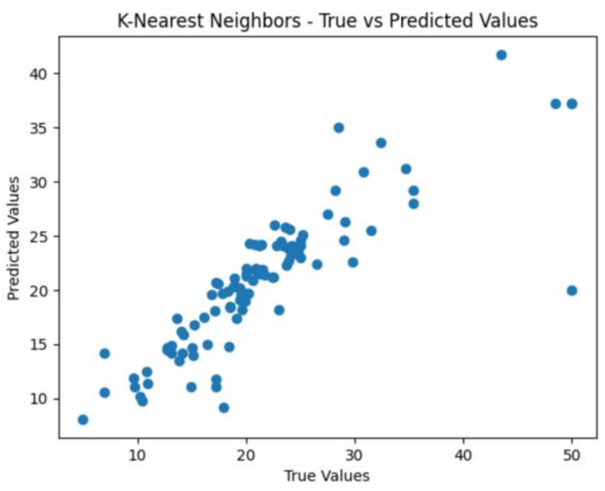
Plotting a scatter plot of true vs. predicted values for each model is a valuable visual technique to assess the performance of a regression model.

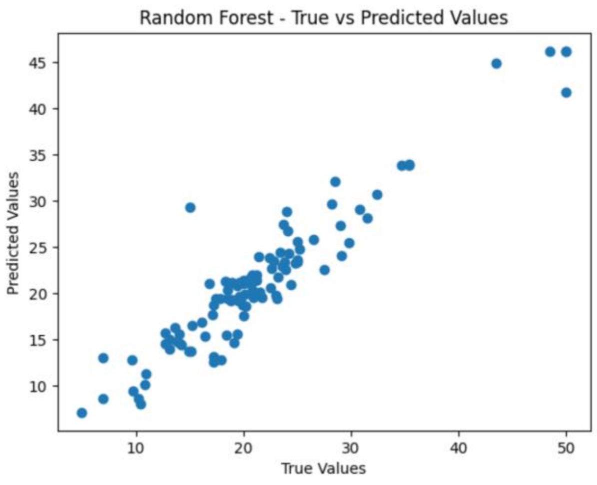
Comparing scatter plots for different models' side by side allows you to visually assess which model performs better. The model with predictions closer to the diagonal line and less spread generally indicates superior performance.



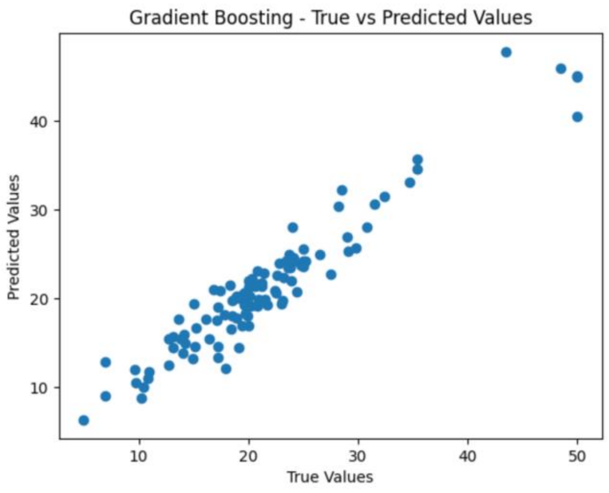

Support Vector Machines - True vs Predicted Values
Predicted Values
True Values
10
15
20
25
30
35
10
20
30
40
50


Ridge Regression - True vs Predicted Values
Predicted Values
True Values
0
10
20
30
40
10
20
30
40
50

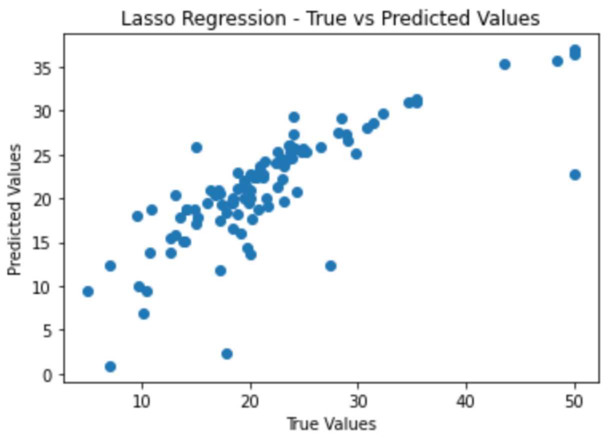

K-Nearest Neighbors - True vs Predicted Values
Predicted Values
True Values
10
15
20
25
30
35
40
10
20
30
40
50



Random Forest - True vs Predicted Values
Predicted Values
True Values



Gradient Boosting - True vs Predicted Values
Predicted Values
True Values



Lasso Regression - True vs Predicted Values
Predicted Values
True Values

CONCLUSION:

In this research paper, we explored the performance of various regression models on the Boston Housing dataset, aiming to predict real estate prices. The evaluation metrics, including Mean Squared Error (MSE) and Explained Variance Score (EVS), provided valuable insights into the strengths and weaknesses of each model.

Among the linear regression models, both Linear Regression and Ridge Regression demonstrated comparable performance, with MSE values around 24.29 and 24.31, respectively. These models explained approximately 66.95% of the variance in the target variable according to the EVS. The regularization applied in Ridge Regression did not significantly impact its performance in this context.

Lasso Regression, while offering regularization, showed a slightly higher MSE of 27.57 and a lower EVS of 0.6245. This suggests that the L1 regularization may have contributed to a reduction in the model's ability to explain variance compared to its non-regularized counterparts.

The Random Forest model emerged as a standout performer, achieving a significantly lower MSE of 7.90 and a high EVS of 0.8927. Random Forest's ensemble nature proved effective in capturing complex relationships within the dataset, making it a robust choice for real estate price prediction.

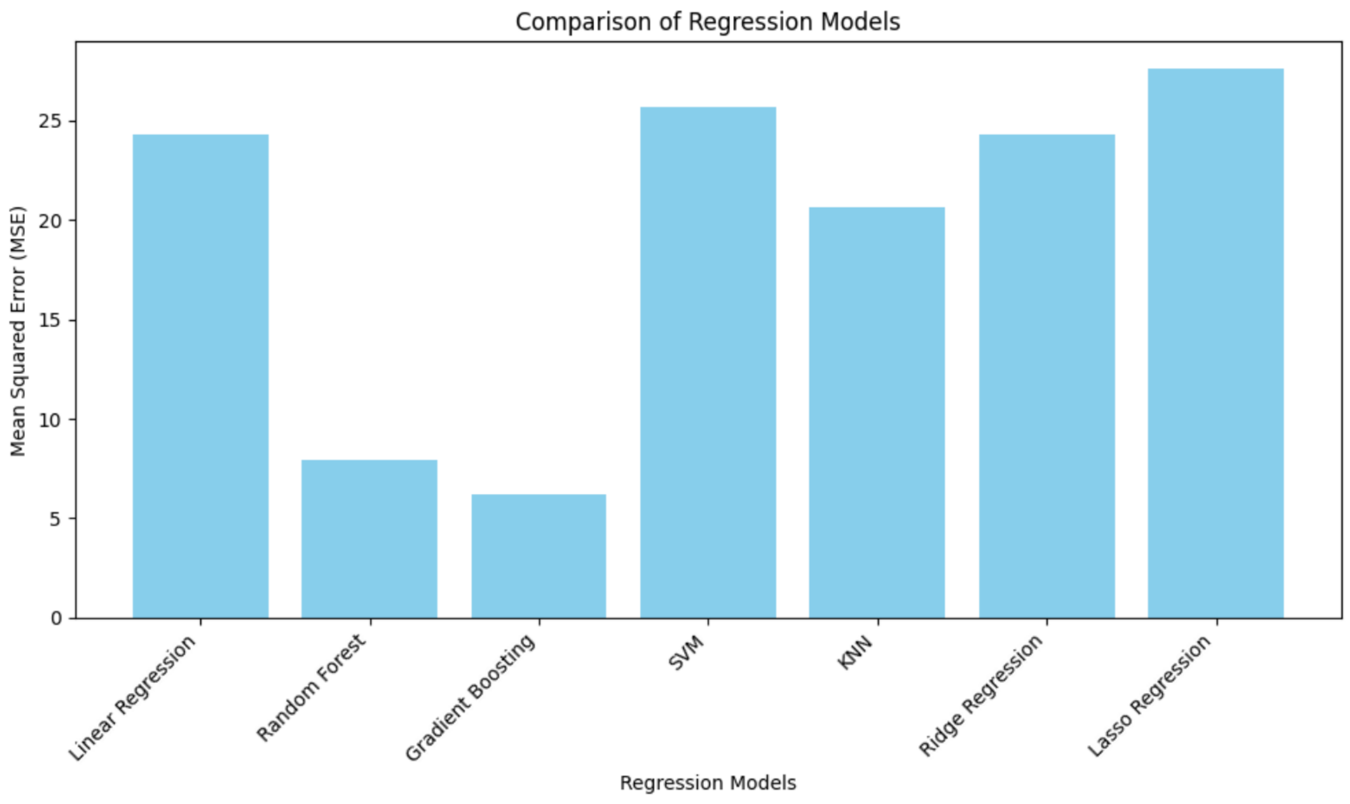
Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) exhibited moderate performance, with SVM achieving an MSE of 25.67 and an EVS of 0.6601, and KNN with an MSE of 20.60 and an EVS of 0.7246. These models fell between the simplicity of linear models and the complexity of ensemble methods.

Notably, Gradient Boosting showcased outstanding performance, achieving the lowest MSE of 6.20 and the highest EVS of 0.9160. Gradient Boosting's sequential correction of errors and ability to capture intricate patterns in the data made it the most effective model in this study.

In conclusion, the choice of a regression model for real estate price prediction in the Boston Housing dataset depends on a balance between accuracy, interpretability, and computational efficiency. While linear models provide a baseline understanding, ensemble methods like Random Forest and Gradient Boosting stand out for their superior predictive performance. This research contributes valuable insights for practitioners seeking to deploy regression models for real estate valuation, considering the trade-offs between model complexity and interpretability. Future research may explore further optimizations, feature engineering, and ensemble strategies to enhance predictive accuracy in real-world applications.

Comparison of all models

```
# List of model names and corresponding predictions
model_names = ['Linear Regression', 'Random Forest', 'Gradient Boosting', 'SVM', 'KNN', 'Ridge Regressi
predictions = [linear_reg_pred, random_forest_pred, gradient_boosting_pred, svm_pred, knn_pred, ridge_p

# Calculate Mean Squared Error for each model
mse_values = [mean_squared_error(y_test, pred) for pred in predictions]

# Plotting the cumulative graph
plt.figure(figsize=(10, 6))
plt.bar(model_names, mse_values, color='skyblue')
plt.xlabel('Regression Models')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('Comparison of Regression Models')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



BIBLIOGRAPHY AND REFERENCES:

1. Yu, Hujia, and Jiafu Wu. "Real estate price prediction with regression and classification." *CS229 (Machine Learning) Final Project Reports* (2016).

2. Ravikumar, A.S., 2017. *Real estate price prediction using machine learning* (Doctoral dissertation, Dublin, National College of Ireland).

3. Mohd, T., Jamil, N.S., Johari, N., Abdullah, L. and Masrom, S., 2020. An overview of real estate modelling techniques for house price prediction. In *Charting a Sustainable Future of ASEAN in Business and Social Sciences: Proceedings of the 3rd International Conference on the Future of ASEAN (ICoFA) 2019—Volume 1* (pp. 321-338). Springer Singapore.

4. https://ieeexplore.ieee.org/document/10117910

5. https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset