Boston Housing Price Prediction

This presentation will delve into predicting median house prices in Boston using various regression techniques, showcasing our findings through Python and data visualization tools.

Exploring the Boston Housing Dataset

An informative look at housing price predictors

- **Goal of the analysis**

  The objective is to build and evaluate ML models for predicting housing prices.

- **Dataset specifics**

  The dataset consists of 506 rows and 13 features, focusing on various housing characteristics.

- **Target variable defined**

  The primary target variable is MEDV, representing median home values in $1000s.

- **Feature diversity**

  Key features include crime rates, zoning data, number of rooms, and property tax rates.

- **Type of problem**

  This analysis falls under supervised regression, aiming to predict continuous values.

- **Importance of dataset**

  It's a classic dataset for understanding factors influencing the real estate market.

Data Preprocessing and Splitting

Understanding Data Preparation for Model Training

- **Data Split into Training and Test Sets**
  The dataset was divided into an 80/20 split for effective training and evaluation.
- **Feature Correlation Analysis**
  Investigated the relationships between features and target variable for insights.
- **Missing Values Check**
  Conducted checks for missing values to ensure data integrity and quality.
- **Normalization of Features**
  Normalized features where necessary to standardize the data for better performance.
- **Visualization with Heatmap**
  Created a heatmap to visualize feature correlations with the target variable MEDV.

Effective Feature Engineering Techniques

- **Combining Related Attributes**

  Merged attributes like TAX and RAD to enhance model accuracy.

- **Scaling Numerical Features**

  Used StandardScaler to normalize numerical features for better model performance.

- **Building Transformation Pipelines**

  Constructed transformation pipelines for a streamlined preprocessing workflow.

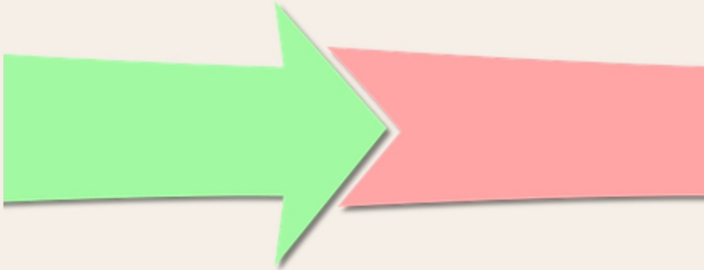
- **Leveraging Scikit-learn**

  Employed scikit-learn's Pipeline to simplify and automate preprocessing steps.

Model Selection: Linear Regression

- **Linear Regression Model Training**
  Trained a Linear Regression model to analyze data effectively.
- **Performance Benchmark with RMSE**
  Achieved moderate RMSE on training data, establishing a solid baseline.
- **Simplicity and Interpretability**
  Linear Regression is simple and interpretable, making results easy to understand.
- **Limitations in Non-linear Patterns**
  This model is limited in capturing complex non-linear patterns in the data.
- **Used as a Performance Benchmark**
  Linear model served as a benchmark for assessing other models' performance.

Understanding Decision Tree Regressor



- **Overfitting Issue**
  The model shows MSE on training as 0, indicating memorization of the training data.
- **Generalization Problem**
  This memorization leads to poor generalization on unseen data, affecting predictions.
- **Visualization Importance**
  Visualizing actual vs predicted values on test data highlights the overfitting problem.

Understanding Cross-Validation Techniques

- **What is Cross-Validation?**
  A technique used to assess how the results of a statistical analysis will generalize to an independent data set.

- **Using k-Fold Cross-Validation**
  We implemented k-fold cross-validation with k=10 for robust model evaluation.

- **Averaging RMSE**
  The Root Mean Square Error (RMSE) was averaged across folds for a reliable performance check.

- **Balancing Overfitting and Underfitting**
  Cross-validation helps in understanding the trade-off between overfitting and underfitting.

- **Realistic Performance Measure**
  Provides a more realistic measure of model performance compared to a simple train/test split.

Final Model Testing Results and Insights

Key findings from the final model evaluation

- **Best-Performing Model Identified**

  The model that delivered the highest performance metrics is highlighted, showcasing its effectiveness for the task.

- **Evaluation on Held-Out Test Set**

  The model was rigorously tested using a separate held-out test set to ensure validity of results.

- **Reported Metrics Overview**

  Key performance metrics such as RMSE, R^2, and MAE are reported to provide a comprehensive understanding of model accuracy.

Model Persistence in Machine Learning

- **Final model saved using joblib**
  The final model is saved using the joblib library for easy access.

- **Quick loading and deployment**
  Joblib allows for quick loading of the model for future use, enhancing deployment efficiency.

- **Predictions without retraining**
  Once loaded, the model can make predictions without needing to be retrained, saving time.

- **Good practice for reproducibility**
  Model persistence is essential for ensuring reproducibility in machine learning workflows.