Boston Housing Prices Prediction

October 9, 2024

1 Introduction

In this assignment, you will work with the **Boston Housing Dataset** to explore the relationships between different features and housing prices. The goal is to understand the dataset through basic exploratory data analysis (EDA), prepare it for a simple machine learning task, and finally build a predictive model using **Linear Regression**. Afterward, you will experiment with different regression models to improve predictions.

Introduction to the Boston Housing Dataset

The Boston Housing Dataset is a classic dataset in machine learning and statistics, widely used for regression tasks. It contains information collected by the U.S Census Service concerning housing prices in the suburbs of Boston, Massachusetts, during the 1970s. The goal of this dataset is to predict the median value of owner-occupied homes, which is the target variable, based on various features that describe the demographic, geographic, and economic aspects of the neighborhoods.

Dataset Features

The dataset consists of 12 features, each representing different characteristics of the Boston suburbs. These features are:

- **CRIM**: Per capita crime rate by town.
- **ZN**: Proportion of residential land zoned for lots larger than 25,000 square feet.
- **INDUS**: Proportion of non-retail business acres per town.
- **CHAS**: Charles River dummy variable (1 if tract bounds river; 0 otherwise).
- **NOX**: Nitric oxide concentration (parts per 10 million).
- **RM**: Average number of rooms per dwelling.
- **AGE**: Proportion of owner-occupied units built before 1940.
- **DIS**: Weighted distances to five Boston employment centers.
- **RAD**: Index of accessibility to radial highways.
- **TAX**: Full-value property tax rate per $10,000$.
- **PTRATIO**: Pupil-teacher ratio by town.
- **LSTAT**: Percentage of lower-status population.

Target Variable

The target variable in this dataset is:

- **MEDV**: Median value of owner-occupied homes in $1000s.

Goal of the Dataset

The primary goal of working with this dataset is to build a model that can predict the median housing prices (`MEDV`) based on the available features. Through this, you will explore how different factors, such as crime rate, number of rooms, and distance to employment centers, impact the price of homes in Boston suburbs. This is a typical regression problem where the task is to fit a model that accurately predicts a continuous target variable.

Part 1: Exploratory Data Analysis (EDA) and Data Cleaning

Before we can build any models, it's essential to understand the dataset, identify any potential issues, and clean the data where necessary. This part will guide you through basic exploratory analysis and data cleaning steps.

Tasks

1. **Load the Boston Housing Dataset**:
   - Use `pandas` to load the data into a DataFrame.
   - Print the first few rows to understand the structure.

   Hint: You can use `pandas.read_csv()` to load the dataset. Here's a starting point:

```
import pandas as pd
df = pd.read_csv("BostonHousing.csv", header=0)
print(df.head())
```

2. **Basic Statistical Analysis**:
   - Calculate summary statistics (mean, median, standard deviation) for each feature.
   - Identify any missing or inconsistent values, and potential outliers that may affect model performance.

   Hint: Use `pandas` methods like `.describe()` to get summary statistics and `.isnull().sum()` to check for missing values.

3. **Data Cleaning**:
   - Check for and handle any missing values. (i.e remove the observations that have missing values)
   - Look for potential outliers in the data, especially in features like `CRIM`, `LSTAT`, and `MEDV`. Outliers can distort the model.

   Hint: You can use visualizations like box plots to detect outliers and `pandas.dropna()` to remove missing values if any exist.

4. **Plotting Distributions**:
   - Plot scatter plots to visualize relationships between `MEDV` and other features (e.g., `RM`, `LSTAT`, etc.).

   Hint: You can use `matplotlib` or `seaborn` for plotting. Here's how to make a simple histogram:

```
import matplotlib.pyplot as plt

df['MEDV'].hist(bins=30)
plt.xlabel('Median Value of Homes')
plt.ylabel('Frequency')
plt.show()
```

Challenge: Use scatter plots to investigate how individual features (like `RM`, the average number of rooms) correlate with `MEDV`.

5. **Correlation Matrix**:
   - Calculate and visualize the **correlation matrix** to see which features are highly correlated with `MEDV` or with each other.

Hint: Use `pandas .corr()` method and `seaborn.heatmap()` for visualizing the correlation matrix:

```
import seaborn as sns

corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

Part 2: Data Preprocessing for Machine Learning

Before applying a machine learning model, we need to preprocess the data. In this section, you will prepare the dataset for a simple **Linear Regression** task.

Tasks

1. **Feature Selection**:
   - Select the features that will be used to predict `MEDV`.
   - You may start by using all features, but consider removing highly correlated or less relevant features after EDA. (eg. remove all features that have less that 0.5 correlation with MEDV)
2. **Splitting the Dataset**:
   - Split the dataset into **training** and **test** sets.

Hint: Use `train_test_split` from `sklearn.model_selection`. Here's an example of how to split your data:

```
from sklearn.model_selection import train_test_split

X = df.drop('MEDV', axis=1)  # Features
y = df['MEDV']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)
```

3. **Standardization (Optional but encouraged)**:
   - Standardize the features using `StandardScaler` from `sklearn.preprocessing`. This ensures that all features are on the same scale.

Hint: Standardization can be done using the `StandardScaler` class in `sklearn`.

Part 3: Building a Linear Regression Model

Now that the data is ready, build a **Linear Regression** model to predict housing prices.

Tasks

1. **Fit a Linear Regression Model**:
   - Train a Linear Regression model using the training data.

   Hint: You can use the `LinearRegression` class from `sklearn.linear_model`:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

2. **Evaluate the Model**:
   - Test the model on the **test set** and calculate performance metrics like **Mean Squared Error (MSE)** or **R-squared (R^2)**.

   Hint: You can use `mean_squared_error` from `sklearn.metrics`:

```
from sklearn.metrics import mean_squared_error

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

3. **Plot the Predictions**:
   - Plot a scatter plot comparing the predicted vs. actual values of `MEDV` in the test set to visually assess model performance.

Part 4: Trying Different Regression Models

Once you've built and evaluated a simple Linear Regression model, it's time to explore other regression techniques. Try the following models from `sklearn` and compare their performance:

Suggested Models

- **Ridge Regression**: Regularized version of Linear Regression, useful when features are highly correlated.
  - **Hint**: Use `Ridge` from `sklearn.linear_model`.
- **Lasso Regression**: Another regularized regression that can help with feature selection by shrinking some coefficients to zero.
  - **Hint**: Use `Lasso` from `sklearn.linear_model`.
- **Decision Tree Regressor**: A non-linear model that splits data into branches and makes decisions at each node.
  - **Hint**: Use `DecisionTreeRegressor` from `sklearn.tree`.
- **Random Forest Regressor**: An ensemble method that combines multiple decision trees to improve predictions.
  - **Hint**: Use `RandomForestRegressor` from `sklearn.ensemble`.

Tasks

1. **Fit and Evaluate Each Model**:
   - Train the above models using the same training data, and evaluate their performance on the test set using metrics like MSE or R^2.
2. **Compare Models**:
   - Compare the performance of the models and identify which model performs best on the test data.

   Challenge: Plot the MSE for each model to visually compare their performance.

Part 5: Reporting Your Findings on GitHub

In this final part of the assignment, you will present your findings by creating a clear and well-structured report in the `README.md` file of a GitHub repository. The emphasis is on documenting your results, insights, and conclusions, while simply uploading the code you've already developed throughout the project.

Tasks

1. **Create a GitHub Repository**:
   - Create a new repository on GitHub with an appropriate name, e.g., `Boston-Housing-Analysis`.
   - Upload all your Python code or Jupyter notebooks used during the project.
   - Ensure the repository contains a `README.md` file, which will serve as the main report for your findings.

   Hint: Follow the official GitHub guide on creating a repository: `https://docs.github.com/en/get-started/quickstart/create-a-repo`.
2. **Document Your Analysis in the `README.md`**:
   - In the `README.md`, provide an overview of your project, including the goal of predicting Boston housing prices using regression models.
   - Summarize the key steps you took, including:
     - Exploratory Data Analysis (EDA)
     - Data Preprocessing
     - Building the Linear Regression model and other regression models
   - Include descriptions of the dataset, such as the most important features and the target variable, to give context.
3. **Present Your Results**:
   - Report the performance metrics of each model you experimented with, such as Mean Squared Error (MSE) and R-squared (R^2), for the test set.
   - Compare the performance of the models (e.g., Linear Regression, Ridge, Lasso, Decision Tree, Random Forest) and discuss which model performed best and why.
   - Provide clear conclusions about the most suitable model for predicting housing prices, with justification based on the results.
4. **Include Visualizations in the `README.md`**:
   - Save any key visualizations you created (scatter plots, correlation matrix, etc.) and upload them to the repository.

- In the `README.md`, embed the visualizations using markdown syntax to illustrate key insights from the data and model performance.
- Provide a brief description for each plot, explaining what it shows and why it's important for understanding the problem or model performance.

Hint: You can include images in markdown using this syntax: `![Alt Text](relative/path/to/image.png)`.

5. **Reflection and Insights**:
   - Reflect on the overall project in the `README.md`, discussing any challenges you encountered and how you addressed them.
   - Mention any interesting findings from the EDA, such as correlations between features and the target variable.
   - Discuss potential improvements or next steps, such as trying additional models or tuning hyperparameters.

Template for the `README.md`

You can structure your `README.md` report as follows:

```
# Boston Housing Analysis

## Project Overview
In this project, we analyze the Boston Housing dataset to predict median
   housing prices using various regression models. The goal is to evaluate
    the relationship between different features and housing prices, and to
    identify the best model for predicting prices.

## Dataset
The Boston Housing dataset contains 13 features and the target variable,
   MEDV (median value of homes). Key features include CRIM (crime rate),
   RM (average number of rooms), and LSTAT (percentage of lower-status
   population).

## Steps
1. Exploratory Data Analysis (EDA)
2. Data Preprocessing
3. Building Linear Regression and other regression models
4. Comparing model performance

## Results
| Model                | MSE   | R-squared |
|----------------------|-------|-----------|
| Linear Regression     | XX.XX | 0.XX      |
| Ridge Regression      | XX.XX | 0.XX      |
| Lasso Regression      | XX.XX | 0.XX      |
| Decision Tree         | XX.XX | 0.XX      |
| Random Forest         | XX.XX | 0.XX      |

**Best Model**: Random Forest, with the lowest MSE and highest R-squared.

## Visualizations
![Scatter Plot](images/scatter_plot.png)
*Figure 1: Scatter plot showing the relationship between RM (average
   number of rooms) and MEDV (housing prices).*
```

```
## Reflection and Insights
During the project, the Random Forest model performed best due to its
    ability to handle complex relationships between features. One key
    finding was the strong positive correlation between RM and MEDV, while
    LSTAT had a negative correlation with housing prices.

## Next Steps
- Tune hyperparameters for the Random Forest model.
- Explore other ensemble models like Gradient Boosting.
```

Final Note: Make sure that your report is clear, concise, and provides sufficient detail for someone to understand your work without needing to look at the code directly.

Final Thoughts

By the end of this part, you will have organized and documented a complete machine learning project on GitHub. This is an essential skill in data science and software development, as it helps in sharing work, collaborating with others, and presenting results in a structured manner.

Resources

- `pandas` Documentation: For data manipulation.
- `matplotlib` Documentation and `seaborn` Documentation: For visualizations.
- `scikit-learn` Documentation: For regression models, train-test split, and performance metrics.

Good luck! Remember to explore the documentation of the libraries mentioned above if you get stuck. Searching for how to use specific functions or methods in Python is a great way to develop your coding skills.