Introduction to Linear Model: an MLR example on Predicting House Prices

A CLA Project

Srimanta Ghosh(2023D005)

1 Introduction

In the realm of statistical modeling, linear models serve as a foundational tool for understanding the relationships between variables. Multiple linear regression, a significant extension, aims to capture the complex interplay between multiple independent variables and a dependent variable. This article delves into the geometric interpretation of linear models through the application of orthogonal projection.

2 Mathematical Formulation

Consider a dataset with n observations and p independent variables denoted as $x_{i1}, x_{i2}, \ldots, x_{ip}$, along with a dependent variable y_i. The multiple linear regression model can be expressed in matrix form as:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon \tag{1}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{2}$$

where $\mathbf{y}$ is an $n \times 1$ vector, $\mathbf{X}$ is an $n \times (p+1)$ matrix (including the intercept term), β is a $(p+1) \times 1$ vector, and ε is an $n \times 1$ vector of error terms.

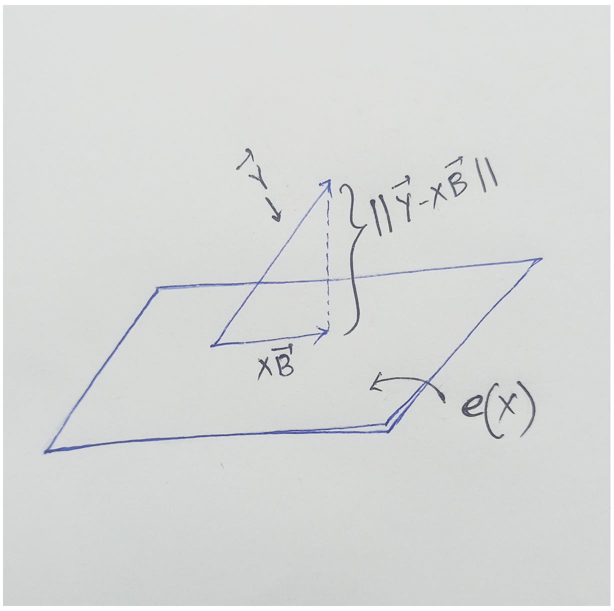
The least squares solution for β is given by:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3}$$

This solution minimizes the Frobenius norm of the residual vector $\mathbf{e} = \mathbf{y} - \mathbf{X}\beta$, representing the discrepancy between observed and predicted values.

3 Orthogonal Projection

The concept of orthogonal projection comes into play when we view the predicted values, $\mathbf{X}\hat{\beta}$, as the projection of $\mathbf{y}$ onto the hyperplane defined by the linear model. The residual vector $\mathbf{e} = \mathbf{y} - \mathbf{X}\hat{\beta}$ is orthogonal to the columns of $\mathbf{X}$, signifying the perpendicular distance between the observed values and the hyperplane.



The orthogonal projection matrix, denoted as $\mathbf{P_X}$, can be expressed as:

$$\mathbf{P_X} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \tag{4}$$

Applying $\mathbf{P_X}$ to $\mathbf{y}$ yields the predicted values:

$$\hat{\mathbf{y}} = \mathbf{P_X}\mathbf{y} \tag{5}$$

The difference vector $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$ captures the orthogonal residuals, emphasizing the minimization of the perpendicular distance between the observed values and the hyperplane.

4 Application: Fitting a MLR Model to a Real Life Dataset(Predicting a House Price)

4.1 Background

The housing market is a complex ecosystem influenced by numerous factors, such as location, size, number of bedrooms, amenities, and economic conditions. Analyzing the interplay of these variables and their impact on house prices requires sophisticated modeling techniques. Linear regression, a fundamental statistical method, offers a clear and interpretable approach to understanding the relationships between independent variables and the target variable, which, in this case, is the house price.

4.2 Objective

The primary objective of this study is to develop a reliable house price prediction model using a linear regression approach. We aim to leverage a dataset containing information about various houses, including their characteristics and historical sale prices. By identifying key features that significantly influence house prices, we seek to build a model that can generalize well to unseen data and provide accurate predictions.

4.3 Dataset Overview

The House Price Prediction Dataset comprises observations of different houses, each characterized by a set of features such as square footage, number of bedrooms, bathrooms, location, proximity to amenities, and other relevant factors. The dataset also includes the corresponding sale prices of these houses, serving as the target variable for our regression model.

```
[1]: Dataset = pd.read_csv("C:/Users/gh22s/Downloads/Housing.csv")
     Dataset.head(10)
```

```
[1]:       price   area  bedrooms  bathrooms  stories mainroad guestroom␣
     ↪basement  \
     0  13300000   7420         4          2        3      yes        no   ␣
     ↪    no
     1  12250000   8960         4          4        4      yes        no   ␣
     ↪    no
     2  12250000   9960         3          2        2      yes        no   ␣
     ↪   yes
     3  12215000   7500         4          2        2      yes        no   ␣
     ↪   yes
     4  11410000   7420         4          1        2      yes       yes   ␣
     ↪   yes
     5  10850000   7500         3          3        1      yes        no   ␣
     ↪   yes
     6  10150000   8580         4          3        4      yes        no   ␣
     ↪    no
     7  10150000  16200         5          3        2      yes        no   ␣
     ↪    no
     8   9870000   8100         4          1        2      yes       yes   ␣
     ↪   yes
     9   9800000   5750         3          2        4      yes       yes   ␣
     ↪    no

       hotwaterheating airconditioning  parking prefarea furnishingstatus
     0              no             yes        2      yes        furnished
     1              no             yes        3       no        furnished
     2              no              no        2      yes   semi-furnished
     3              no             yes        3      yes        furnished
     4              no             yes        2       no        furnished
     5              no             yes        2      yes   semi-furnished
     6              no             yes        2      yes   semi-furnished
     7              no              no        0       no      unfurnished
     8              no             yes        2      yes        furnished
     9              no             yes        1      yes      unfurnished
```

4.4 Observing Categorical Features

Categorical features like location, amenities, and other non-numeric attributes provide valuable insights into the dataset.

[2]:
```
categorical_features = Dataset.select_dtypes(include = ["object"]).
 ↪columns
categorical_features
```

[2]:
```
Index(['mainroad', 'guestroom', 'basement', 'hotwaterheating',
       'airconditioning', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

4.5 Observing Numerical Features

An initial exploration involves observing numerical features such as square footage, number of bedrooms, bathrooms, and other quantitative characteristics.

[3]:
```
numerical_features = Dataset.select_dtypes(exclude = ["object"]).
 ↪columns
numerical_features
```

[3]:
```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking'],
dtype='object')
```

4.6 Removing Categorical Features

As linear regression assumes a linear relationship between variables, we temporarily remove categorical features for model simplicity. Later, advanced techniques can be explored to incorporate these features.

[4]:
```
DS = Dataset.select_dtypes(exclude = ["object"])
DS.head()
```

[4]:
```
      price  area  bedrooms  bathrooms  stories  parking
0  13300000  7420         4          2        3        2
1  12250000  8960         4          4        4        3
2  12250000  9960         3          2        2        2
3  12215000  7500         4          2        2        3
4  11410000  7420         4          1        2        2
```

4.7 Checking for Null Values

Ensuring data completeness is crucial. We investigate the dataset for null values that might affect the accuracy of our linear regression model.

[5]:
```
DS.isna().sum()
```

[5]:
```
price        0
area         0
bedrooms     0
bathrooms    0
stories      0
parking      0
dtype: int64
```

4.8 Dependent Feature (Actual Price)

We visualize the distribution of house prices, our dependent variable, to understand its characteristics.

[3]:
```
Y=DS['price']
dfY = pd.DataFrame({"price":Y})
dfY
```

[3]:
```
        price
0    13300000
1    12250000
2    12250000
3    12215000
4    11410000
..        ...
540   1820000
541   1767150
542   1750000
543   1750000
544   1750000

[545 rows x 1 columns]
```

4.9 Independent Features

Numerical features are displayed individually to observe their distributions and relationships with the dependent variable.

[4]:
```
drop_price = ['price']
X=DS.drop(drop_price,axis=1)
X
```

[4]:
```
     area  bedrooms  bathrooms  stories  parking
0    7420         4          2        3        2
1    8960         4          4        4        3
2    9960         3          2        2        2
3    7500         4          2        2        3
4    7420         4          1        2        2
..    ...       ...        ...      ...      ...
540  3000         2          1        1        2
541  2400         3          1        1        0
542  3620         2          1        1        0
543  2910         3          1        1        0
544  3850         3          1        2        0

[545 rows x 5 columns]
```

4.10 Design Matrix

The design matrix **X**, formed by combining independent features, is displayed. This matrix is essential for model training and prediction.

```
[8]: col1 = {'Column': [1] * 545}
     col_1 = pd.DataFrame(col1)
     Design_matrix = np.concatenate((col_1,X.values), axis=1)
     print(Design_matrix)
```

```
[[   1 7420    4    2    3    2]
 [   1 8960    4    4    4    3]
 [   1 9960    3    2    2    2]
 ...
 [   1 3620    2    1    1    0]
 [   1 2910    3    1    1    0]
 [   1 3850    3    1    2    0]]
```

4.11 Estimating β

The heart of linear regression lies in estimating β that define the relationship between independent and dependent variables. We'll delve into the estimation process and interpret the significance of each coefficient.

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

```
[9]: x = Design_matrix
     x_t = x.T
     beta_cap = np.linalg.inv(x_t @ x) @ x_t @ Y
     dfbeta = pd.DataFrame({"$\hat{\u03B2}$":beta_cap})
     dfbeta["$\hat{\u03B2}$"]=dfbeta["$\hat{\u03B2}$"].round().
      ↪astype('int64')
     dfbeta
```

```
[9]:    $\hat{β}$
     0    -145734
     1        331
     2     167810
     3    1133740
     4     547940
     5     377596
```

4.12 Projection Matrix (Predicted Price)

The projection matrix, obtained by applying the estimated coefficients to the design matrix, provides a set of predicted house prices. Visualizing the projection matrix enables a comparison between actual and predicted prices. $\mathbf{P_X}$ for predicting house prices:

$$\mathbf{P_X} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

```
[10]: Projection_matrix = x @ np.linalg.inv(x_t @ x) @ x_t @ Y
      dfproj_M = pd.DataFrame({"Projection Matrix(Predicted Price)":
       ↪Projection_matrix})
      dfproj_M["Projection Matrix(Predicted Price)"]=dfproj_M["Projection␣
       ↪Matrix(Predicted Price)"].round().astype('int64')
      dfproj_M
```

```
[10]:      Projection Matrix(Predicted Price)
      0                               7648874
      1                              11351808
      2                               7774158
      3                               7505020
      4                               5967194
      ..                                  ...
      540                             3620104
      541                             2834052
      542                             3070203
      543                             3002921
      544                             3862109

      [545 rows x 1 columns]
```

4.13 Comparing Actual and Predicted Price

By comparing the actual and predicted house prices, we can assess the model's performance. Visualization aids in identifying patterns and discrepancies, allowing for model refinement.

```
[11]: dfcmpsN = pd.DataFrame({'Actual': Y, 'Predicted': Projection_matrix})
      dfcmpsN['Predicted']=dfcmpsN['Predicted'].round().astype('int64')
      dfcmpsN
```

```
[11]:        Actual  Predicted
      0    13300000    7648874
      1    12250000   11351808
      2    12250000    7774158
      3    12215000    7505020
      4    11410000    5967194
      ..        ...        ...
      540   1820000    3620104
      541   1767150    2834052
      542   1750000    3070203
      543   1750000    3002921
      544   1750000    3862109

      [545 rows x 2 columns]
```

4.14 MSE

Evaluating the model's performance by calculating the Mean Squared Error (MSE) as a measure of prediction accuracy:

$$\text{MSE} = \frac{1}{n}\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$$

```
[12]: from sklearn import metrics
      print('Mean Absolute Error:',
            metrics.mean_absolute_error(Y, Projection_matrix))
```

```
Mean Absolute Error: 909001.052585545
```

4.15 Calculating House Price using User Input Data from the Fitted Model

Finally, we demonstrate how the fitted linear regression model can be utilized to predict house prices based on user-provided input data.

[13]:
```
const = 1
AREA = int(input("AREA = "))
BEADROOMS = int(input("No. of Beadrooms = "))
BATHROOMS = int(input("No. of Bathrooms = "))
STORIES = int(input("No. of Stories = "))
PARKING = int(input("No. of Parkings = "))
casual = np.array([[const,AREA,BEADROOMS,BATHROOMS,STORIES,PARKING]])

Price_emt = beta_cap @ casual.T
print("Price_Estimated={}".format(Price_emt[0]))
```

```
AREA = 5000
No. of Beadrooms = 4
No. of Bathrooms = 6
No. of Stories = 4
No. of Parkings = 2
Price_Estimated=11930474.930881787
```

4.16 Significance of Linear Regression

Linear regression is a widely employed technique in the field of predictive modeling due to its simplicity, interpretability, and effectiveness in capturing linear relationships between variables. In the context of house price prediction, a linear regression model allows us to estimate the impact of each independent variable on the target variable, providing valuable insights into the factors influencing property values.

4.17 Post-Project Analysis: Identifying Future Focus Areas for Continuous Improvement

As an extension, we can explore the ANCOVA (Analysis of Covariance) model—a refinement over linear regression that incorporates categorical variables. This model offers a more comprehensive approach to capturing the nuanced factors influencing house prices.

5 Conclusion

In conclusion, the matrix-based formulation of multiple linear regression, when viewed through the lens of orthogonal projection, offers a robust and geometrically intuitive approach to parameter estimation. The least squares solution not only minimizes the sum of squared differences but also establishes a connection between linear models and the concept of projecting observations onto a hyperplane defined by the model.

As we delve into the application of linear regression in predicting house prices, we anticipate uncovering patterns and relationships within the dataset that contribute to accurate price predictions. By understanding the nuances of the housing market through statistical modeling, we aim to provide a valuable tool for stakeholders to make informed decisions in the dynamic and competitive real estate landscape.