Machine Learning

Implementing Linear Regression classifier

Contents

1 Objectives

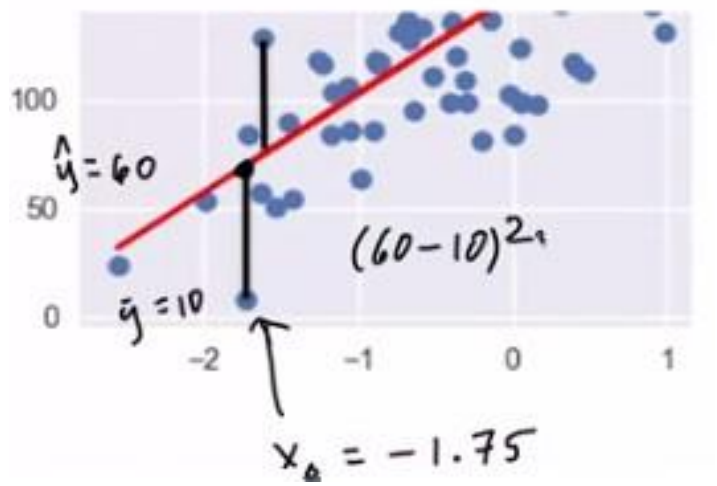
- To understand the cost function for Linear regression
- To know how to find the coefficients for Linear Regression
- To minimize the cost function using gradient descent
- To plot the results obtained through numerical, library and polyfit operations in python

2 Problem Discussion

Regression analysis is used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. Besides, regression analysis is widely used for prediction, forecasting, and classification where its use has considerable connect with the field of machine learning. In this lab you are required to implement the Linear Regression classifier using the python library, the numerical method technique and lastly the polyfit built in function.

2.1 Least Squares Technique

Linear regression is based on Least Square Estimation which says regression coefficients should be chosen in such a way that it minimizes the sum of the squared distances of each observed response to its fitted value. A linear model is a sum of weighted variables that predicts a target output value given an input data instance.



2.2 Numerical Method

The task is to find a line which fits best in the scatter plot so that we can predict the response for any new feature values. (i.e a value of x not present in dataset). This line is called regression line.
The equation of regression line is represented as:

$$h(x_i) = \beta_0 + \beta_1 x_i$$

Here,

- h(x_i) represents the predicted response value for ith observation.
- b_0 and b_1 are regression coefficients and represent y-intercept and slope of regression line respectively.

To create our model, we must "learn" or estimate the values of regression coefficients b_0 and b_1. And once we've estimated these coefficients, we can use the model to predict responses! In this article, we are going to use the Least Squares technique. So, our aim is to minimize the total residual error.
We define the squared error or cost function, J as:

$$J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} \varepsilon_i^2$$

And our task is to find the value of b_0 and b_1 for which J(b_0,b_1) is minimum! Without going into the mathematical details, we present the result here:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

where SS_xy is the sum of cross-deviations of y and x:

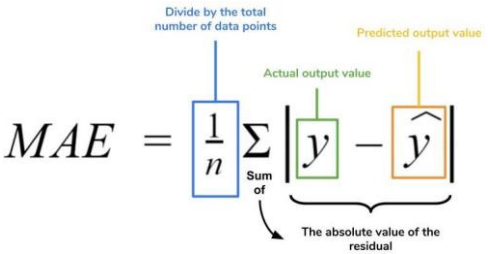
$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y}$$

and SS_xx is the sum of squared deviations of x:

$$SS_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n(\bar{x})^2$$

2.3 Error Metrics: MAE, MSE, R2 score

The mean absolute error (MAE) is the simplest regression error metric to understand. We'll calculate the residual for every data point, taking only the absolute value of each so that negative and positive residuals do not cancel out. We then take the average of all these residuals.



The mean square error (MSE) is just like the MAE, but squares the difference before summing them all instead of using the absolute value. We can see this difference in the equation below.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)}^{2}$$

The square of the difference between actual and predicted

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

```
y_residue = y - ypred
y_tot = y - np.mean(y)
sy = sum(pow(y_residue,2))
ss = sum(pow(y_tot,2))
r2val = 1 - (sy/ss)
```

2.4 Polyfit and Polyval

The method polyfit measures the coefficients for n-degree polynomial. This method can be written as:

p = polyfit(x,y,n)

Here,

p will store the coefficents.

n = degree of polynomial

x = independent variable

y = dependent variable

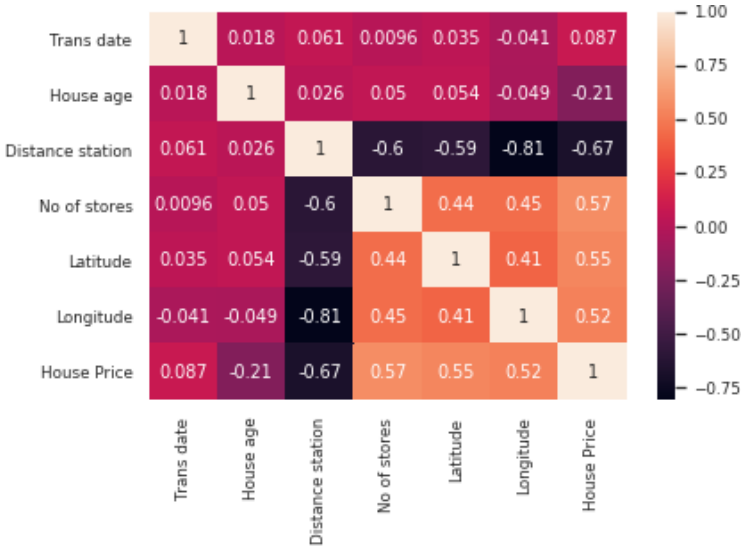
In order to get the predicted y a method known as polyval is used. It takes the p(obtained from polyfit) and x as parameters.

yp = polyval(p,x)

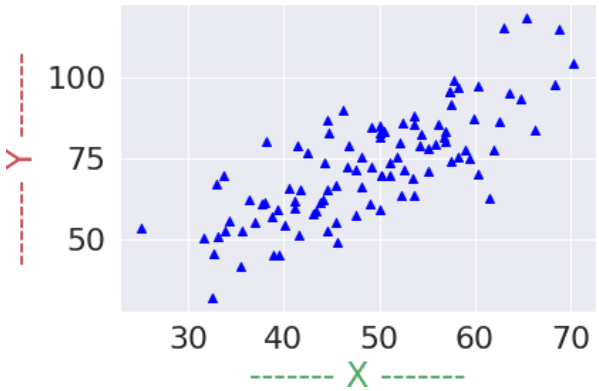
```
p = polyfit(x,y,1)
yp_pred = polyval(p,x)
```

3 Tasks

- **Task 01:**
  Load the dataset and correlate it to find two most strongly correlated features. [Hint: use corr() method for correlation]



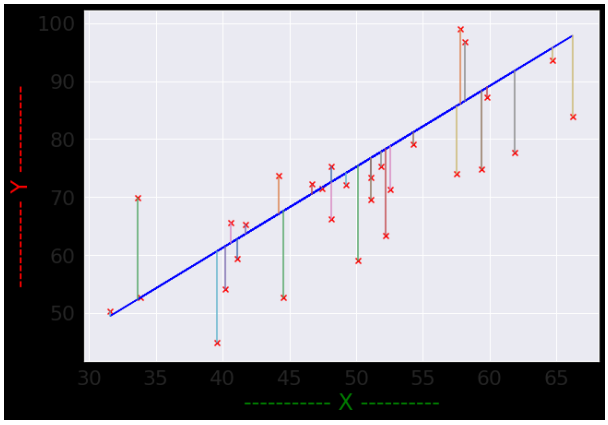
- **Task 02:**
  Extract the two features from the previous and plot them using scatterplot.



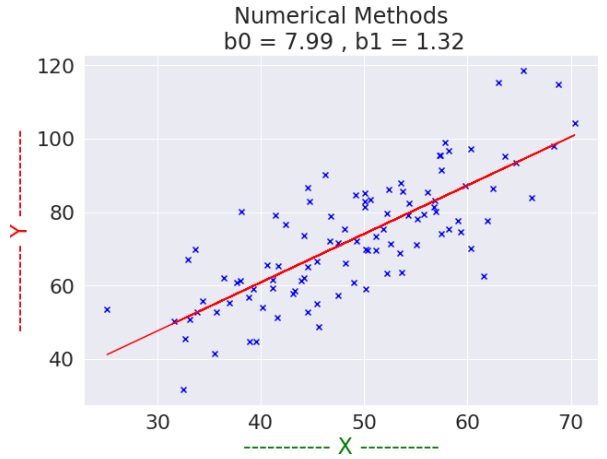
- **Task 03:**
  Use those two features to split the dataset into training and testing sets and implement the Linear Regression classifier from the python library. Besides this, display the coefficient, intercept(b_0, b_1) and the test accuracy for this model. [Hint: Use coef_ and intercept_]

- **Task 04:**
  Display the mean absolute error (MAE), mean squared error (MSE) and R2-score for this model using built in function.

- **Task 05:**
  Plot the actual test data(xtest,ytest) in the form of a scatter plot and the fitted line for the predicted data(xtest,ypred) over it for this model. Also display the residuals(line from the actual data onto the predicted line) and its distance. [Hint: Use plot function for plotting a straight line and use Euclidean distance)

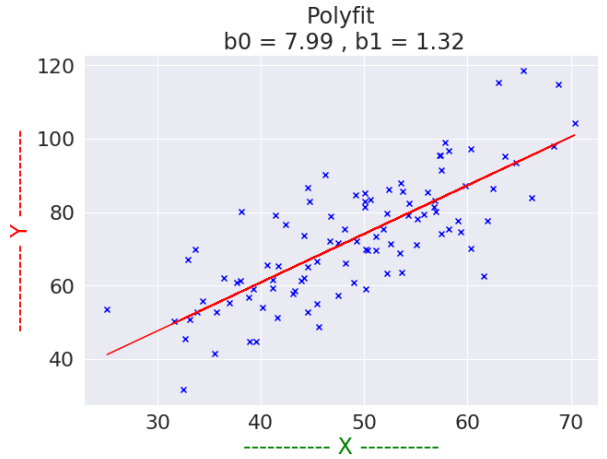


- **Task 06:**
  Implement the Linear Regression model on the dataset using numerical methods and display its coefficient and intercept(b_0, b_1). Display the best fitted line over the actual data as well.



- **Task 07:**
  Display the mean absolute error (MAE), mean squared error (MSE) and R2-score for the numerical method approach without using any built in function.

- **Task 08:**
  Implement the Linear Regression model on the dataset using the polyfit function and display its coefficient and intercept(b_0, b_1). Display the best fitted line over the actual data as well.



- **Task 09:**
  Compare the best fitted line for the above mentioned methods. [Hint: Use subplot]